



freeBSDTM

JOURNAL

Nov/Dec 2014

Vol. 1 • Issue No. 6

Battles of the
Graphics Stack:

VIDEO DRIVERS

pfSense:

A 3-Hour Tour

**No More
Printfs:**

Digging into the
Kernel with DTrace

**NEW
COLUMN!**

Book Review

(pfSense)¹⁰

We're taking pfSense®
to a higher power.



Secure your network with pfSense®2.2 and FreeBSD®10 on the
new Netgate®C2758.

With 8 Intel® processing cores, up to 8 GbE ports, AES-NI and multi-threaded
packet filtering, we're advancing network security to a higher power.



7212 McNeil Drive Suite 204 Austin, TX 78729 +1.512.646.4100

Netgate® is a registered trademark of Rubicon Communications, LP. pfSense® is a registered trademark of Electric Sheep Fencing, LLC.
Intel is a trademark of Intel Corporation in the U.S. and/or other countries.

Table of Contents

FreeBSD Journal

Vol. 1, Issue No. 6



3 Foundation Letter

The FreeBSD Foundation has received a \$1 million donation from Jan Koum CEO and cofounder of WhatsApp. This marks the largest single donation to the Foundation since its inception almost 15 years ago and serves as another example of someone using FreeBSD to great success and then giving back to the community. *By FreeBSD Journal Board*

32 Book Review

Focusing on SSL since 2009, and dedicating two years to writing, including six months of rewriting and chapter expansion, Ivan Ristic has released his second book, *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications*. *By Steven Kreuzer*

34 Ports Report

New talent enters the ranks of ports committers, noteworthy changes to the ports tree, important ports updates—and more. *By Frederic Culot*

36 Conference Report

The Grace Hopper Celebration of Women in Computing is the world's largest gathering of women technologists. *By Shteryana Shopova*

38 svn update

Thanks to the FreeBSD community and the FreeBSD developers for all the hard work that went into the FreeBSD 10.1-RELEASE! *By Glen Barber*

40 This Month in FreeBSD

FreeBSD turned 21 this year as the first RELEASE announcement went out on November 1, 1993. This month's column takes a look at some of the features of the 10.1-RELEASE. *By Dru Lavigne*

42 Events Calendar

By Dru Lavigne

Battles of the Graphics Stack:

VIDEO DRIVERS

4 THE GRAPHICS STACK RESPONSIBILITIES EXTEND WELL BEYOND THE LITERAL “GRAPHICS” WORLD. WHERE DOES FREEBSD STAND IN THIS COMPLEX WORLD AND WHAT ARE THE CHALLENGES AHEAD OF US?

By Jean-Sébastien Pédron

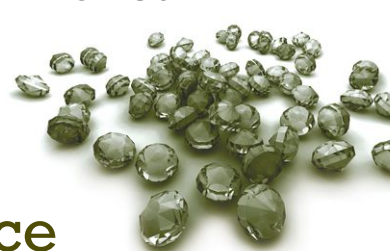
No More Printf's: Digging into the Kernel with DTrace

12 DTrace is a useful tool for gaining visibility into the inner workings of the FreeBSD kernel, but, as with any tool, DTrace has its share of limitations and problems. It is important to be aware of them and know how to work around them. *By Mark Johnston*

pfSense: A 3-Hour Tour

22 pfSense® software makes FreeBSD firewalls accessible to every IT professional capable of managing a typical commercial-grade firewall. Release 2.2 is the culmination of 15 months of development effort and this article highlights the most significant changes.

By Chris Buechler and Jim Thompson



Help Grow
FreeBSD
(Donate Today...)

Your donations over the past year have led to great progress for FreeBSD and the community.

Thank you!!

The Foundation is looking forward to continuing that progress in 2015. The areas we'd like to invest in include the following:

Funding Projects to Advance FreeBSD

Projects may include:

- Enhancements to the binary package build, distribution, and verification mechanism.
- Improving automated testing.
- Updated development and performance analysis tools.
- Support for the 64-bit ARM architecture.

Increasing Our FreeBSD Marketing Efforts

- Creating additional marketing collateral to recruit for FreeBSD
- Providing more marketing literature to educate on FreeBSD
- Helping grow our base of community investors
- Promoting the Project and The Foundation

Providing Conference Resources and Travel Grants

- Facilitating a greater number of FreeBSD representatives in attending conferences around the world to give presentations on FreeBSD.
- Providing FreeBSD materials in multiple languages



Support FreeBSD

Donate to the Foundation!

FreeBSD is internationally recognized as an innovative leader in providing a high-performance, secure, and stable operating system. Our mission is to continue and increase our support and funding to keep FreeBSD at the forefront of operating system technology.

Thanks to people like you, the FreeBSD Foundation has been proudly supporting the FreeBSD Project and community for almost 15 years. We are incredibly grateful for all the support we receive from you and so many individuals and organizations that value FreeBSD.

Make a gift to support our work in 2015. Help us continue and increase our support of the FreeBSD Project and community worldwide!

Making a donation is quick and easy. Go to:
www.freebsdoundation.org/donate



The
FreeBSD
FOUNDATION

www.freebsdoundation.org



FreeBSDTM JOURNAL Editorial Board

- John Baldwin • Member of the FreeBSD Core Team
- Daichi Goto • Director at BSD Consulting Inc. (Tokyo)
- Joseph Kong • Author of *FreeBSD Device Drivers*
- Dru Lavigne • Director of the FreeBSD Foundation and Chair of the BSD Certification Group
- Michael W. Lucas • Author of *Absolute FreeBSD*
- Kirk McKusick • Director of the FreeBSD Foundation and lead author of *The Design and Implementation* book series
- George Neville-Neil • Director of the FreeBSD Foundation and co-author of *The Design and Implementation of the FreeBSD Operating System*
- Hiroki Sato • Director of the FreeBSD Foundation, Chair of AsiaBSDCon, member of the FreeBSD Core Team and Assistant Professor at Tokyo Institute of Technology
- Robert Watson • Director of the FreeBSD Foundation, Founder of the TrustedBSD Project and Lecturer at the University of Cambridge

S&W PUBLISHING LLC
PO BOX 408, BELFAST, MAINE 04915

- Publisher** • Walter Andrzejewski
walter@freebsdjournal.com
- Editor-at-Large** • James Maurer
jmaurer@freebsdjournal.com
- Art Director** • Dianne M. Kischitz
dianne@freebsdjournal.com
- Office Administrator** • Cindy DeBeck
cindy@freebsdjournal.com
- Advertising Sales** • Walter Andrzejewski
walter@freebsdjournal.com
Call 888/290-9469

FreeBSD Journal (ISBN: 978-0-615-88479-0) is published 6 times a year (January/February, March/April, May/June, July/August, September/October, November/December).

Published by the FreeBSD Foundation,
PO Box 20247, Boulder, CO 80308
ph: 720/207-5142 • fax: 720/222-2350
email: board@freebsdjournal.org
Copyright © 2014 by FreeBSD Foundation.
All rights reserved.

This magazine may not be reproduced in whole or in part without written permission from the publisher.

LETTER from the Board •

Welcome to our ultimate issue for 2014. Before going any further, we want to thank each and every subscriber for demonstrating your belief in the mission of the *FreeBSD Journal* by joining us in our first year of publication. We hope you'll come along for the ride in 2015 as we'll continue to publish all that's best in the FreeBSD world.

Before pointing out some of the highlights in this issue, we want to convey some very exciting news. The FreeBSD Foundation recently received a \$1 million donation from Jan Koum, CEO and cofounder of WhatsApp. This marks the largest single donation to the Foundation since its inception almost 15 years ago and serves as yet another example of someone using FreeBSD to great success and then giving back to the community. Needless to say, all of us who work on the *Journal* also work on FreeBSD and we're overwhelmed by the generosity of this donation.

This issue contains three excellent articles on three very different topics. One of the challenges for any open-source operating system is keeping up-to-date with video drivers. Video hardware is complex to program, and the documentation of such systems is often a closely held secret by the companies that produce the chips. Jean-Sébastien Pédron leads us through the graphics stack in FreeBSD to help us better understand what is going on under the root weave screen.

One of the best ways, and often the only way, to understand complex software is to have some level of transparency into what is happening at run time in a live system. The DTrace system in FreeBSD is the premier tool for achieving the level of transparency needed to understand and debug such complex issues. Mark Johnston, who has been doing a great deal of work on this topic, explains how to use DTrace where—in the past—programmers would have had to use the venerable `printf()` function to determine what's going on in the guts of the system.

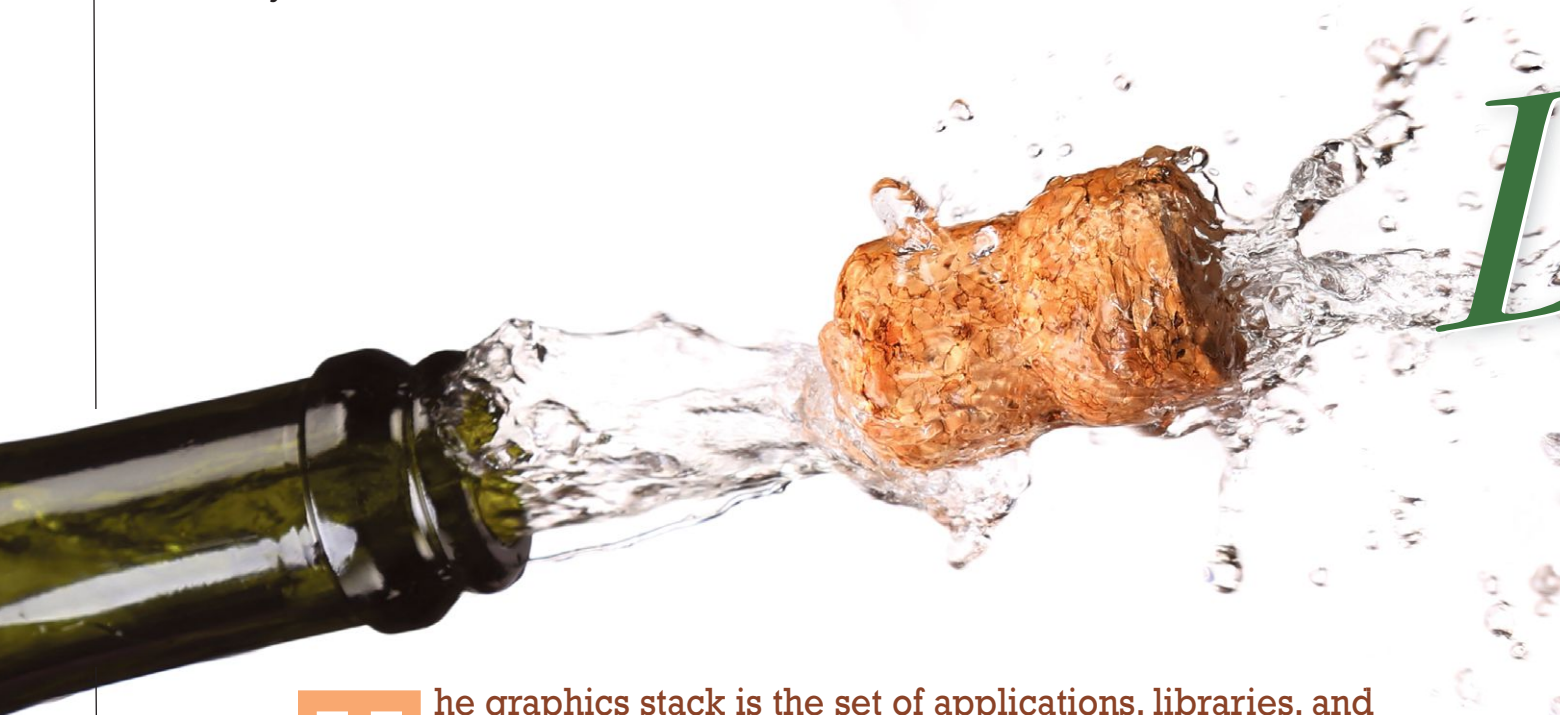
While Linux has its distros, in the FreeBSD world we try to create a full system that includes all the necessary tools to get a job done. That doesn't mean that a FreeBSD system is the be-all and end-all of any software system, but it is a complete base on which to build a product. pfSense® is a great example of software that both modifies and extends FreeBSD to create a unique product, a complete firewall, that can be applied in both large enterprises and in small offices. Chris Buechler and Jim Thompson take us through the latest release of pfSense to show us how it's done and how open-source software, with targeted improvements and extensions, can be used to build a commercial-grade firewall.

There are many holidays at the end of the calendar year, but there will be no rest for the *Journal's* editorial board. We just completed a face-to-face editorial planning meeting following the MeetBSD Conference in Silicon Valley, and we're all excited by what's in store for 2015. It's amazing what a focused group of smart people can get done in two hours of in-person discussion. Topics for 2015 include: Storage and Filesystems, Services on Top of FreeBSD, FreeBSD in the Cloud, and, of course, FreeBSD 11, which is due out next year. We're looking forward to an exciting array of articles and columns in 2015, and we hope you are as well. Thanks to the authors, columnists, board members, the FreeBSD Foundation, and especially the *FreeBSD Journal* readers, who have made our first year of publication so successful.

Sincerely,
FreeBSD Journal Editorial Board

BATTLES OF THE GRAPHICS STACK:

By Jean-Sébastien Pédron



The graphics stack is the set of applications, libraries, and drivers that allow you to display a desktop on your workstation or to render the beautiful game you were about to close two hours ago—that's the most obvious role we all think of. Other less obvious roles are, for example, offering the possibility to run computational tasks on the GPU ("GPGPU" for general-purpose computing on graphics processing units), handling all the various input devices, or even providing the tools and mechanisms required to make a computer accessible to anyone. Thus the graphics stack responsibilities extend well beyond the literal "graphics" world.

VIDEO DRIVERS



Where does FreeBSD stand in this complex world? What are the challenges ahead of us? Let's start with a glass of fine Pauillac!

The X.Org Developer's Conference 2014

"Could you pass me the Château Franc Cros?"

"This Château Altimar is very nice!"

"How do you pronounce that?"

"Cheers!"

It's Friday evening, October 10th, and we are at "Aux 4 coins du vin" in Bordeaux, France, tasting some really great white and red wines and talking about the past three days of the conference, but more importantly, having a wonderful time together!

This wine tasting event and the visit of Saint-Émilion the next day were closing the 2014 X.Org Developer's Conference (XDC). The conference which is held every year and alternates between Europe and North America, is the meeting place for all people

interested in the free open-source graphics stack. This year's XDC took place October 8 to 10 at the University of Bordeaux in south-west France.

Talks were very interesting and covered topics such as input handling and accessibility, progress in all video drivers, recent changes in X.Org and Wayland, and how to improve security. And for the first time there was a BSD track! The program is available online and includes links to slides and videos of the talks.

I had the privilege of attending this confer-

APPLICATIONS

X.Org server

X.Org's DDX

Mesa driver

Kernel video driver

ence and representing the FreeBSD Project. On Friday morning, François Tigeot, Matthieu Herrb, and I presented the status of the graphics stack on DragonFlyBSD, OpenBSD, and FreeBSD, respectively. But my main goal was to meet the people behind the graphics stack and show that FreeBSD still cares about it.

This was needed for two reasons:

First, a working graphics stack is critical for keeping longtime FreeBSD users and also for attracting new blood, because FreeBSD can run on its workstation. Even if FreeBSD does not target the “desktop market,” junior sysadmins tend to deploy the same operating system on their production machines as the one on their workstations—primarily because they become comfortable with it. A sysadmin may also want to install FreeBSD on an unused computer, just to play with it a bit before migrating all the web servers. Longtime users' habits are hard to break: it's always nice to use the same tools on the laptops as on the servers.

Second, a working graphics stack is critical for the new trend of using the GPU's high power and parallelism to perform computational tasks. And it's not only useful for mining virtual currency locally; it's also good for mining virtual currency on large clusters too.

Today, most of the development on the graphics stack is done by the Linux community and companies investing in Linux, so they rightfully target Linux only—either they are paid to do so, or they do it for fun in their spare time. I'm happy to report that the goal of reviving the relationship with X developers was a great success! People were very kind and welcoming, they showed interest in BSD in general, and even offered their help. Now it's our responsibility to get to work!

Video Drivers

Among the many challenges ahead of us, video drivers are at the heart. Drivers are divided into

three parts: the kernel, Mesa, and X.Org's “DDX” (`xf86-video-*`).

Because there is so much to say about drivers, the scope of this article is limited to some of the issues we need to solve in the kernel and with Mesa. Future articles will talk about other aspects as other problems are being worked on.

To set the stage, this article provides an overview of the concepts. A report of the XDC is available on the [graphics team blog](#), thus I won't repeat that here.

Kernel Video Drivers

Why kernel drivers? Starting with the FreeBSD 9.1-RELEASE and the i915 GPU in 2012, video drivers were moved from userland to the kernel. In 2013, the driver for Radeon GPUs was added. This was a major turnaround after 25 years of the graphics stack being entirely in userspace. Why this sudden change of direction?

At the time of XFree86, which was later forked to become the X.Org project, the X server was a monster—it performed the same tasks and provided the same services generally attributed to a kernel. It scanned buses, drove input and video devices, managed video memory, and tried to address security concerns. The X Window System predates Linux or any BSD. The purpose of handling everything was to guarantee portability across all UNIX-like operating systems.

Unfortunately, this design had many drawbacks:

- The code base was huge and very hard to maintain. Furthermore, a large part of the services was already provided by the kernel.
- Both the kernel and the X server managed buses and devices. For instance, this made suspend/resume impossible to implement properly.
- To be able to do this, the X server ran with root privileges. The `xorg` executable is setuid to let a normal user start an X session.
- The performance was low, mainly because of all the required communication between userspace and kernelspace.

- GPGPU was not possible without running an X server, because it managed the hardware itself.
- Drivers could not be reused by another project, like Wayland.
- Using a GPU in a virtual environment was impossible.
- Switching between the console and X, either when starting/exiting an X session or when using **Ctrl-Alt-Fx** caused screen flickering.
- This often prevented the kernel debugger from working properly, meaning that obtaining a core dump was unreliable.

During this era, an optional subsystem, called the Direct Rendering Manager (DRM) appeared, and it contained a small portion of the drivers. In FreeBSD, the code is in `sys/dev/drm`. This early DRM was an optimization, but didn't resolve those problems, as all the important code remained in X.Org DDxs.

Moving the GPU drivers entirely to the kernel DRM subsystem solves all of them. However, it puts the effort on the kernel developers, and, without constant maintenance, the level of support varies between operating systems. Fortunately, the drivers kept their MIT/X11 license, allowing us to take Linux drivers and port them to FreeBSD.

The DRM Subsystem

DRM offers an API to userland applications for sending commands and data to a GPU and for reading data back from it. It's comprised of the following components:

- DRM itself, which provides device-independent functionalities for both userland applications and other components of this subsystem.
- TTM and GEM are two memory managers. They handle objects in system and video memories, movements between them, and memory mapping, paging, and swapping.
- The drivers themselves. We have two at the time of this writing: the i915 driver, which relies on the GEM memory manager, and the Radeon driver, which depends on TTM.

Maintaining DRM

In FreeBSD, the initial port of the GEM memory manager and the i915 driver was based on a copy of the old DRM subsystem located in `sys/dev/drm2`. Missing device-independent code was added as required and the original Linux code was migrated to use the FreeBSD native facilities (memory allocation, locking primitives, I²C API, device management API, etc.). When the TTM memory manager and the Radeon driver were ported, the same method was used.

On the plus side, the code looks familiar to a FreeBSD developer and thus can be more easily understood. On the downside, this penalizes the maintenance of the DRM subsystem, because new code from Linux must be ported again.

Drivers are critical in the graphics stack. We deeply need an up-to-date DRM subsystem because it is required to:

- support newer hardware, such as Intel Haswell GPU, which is almost ready to be added to FreeBSD,
- provide new services such as DRM PRIME, which permits the movement of data between multiple GPUs and userspace.

Today, we have a hard time keeping this subsystem on par with Linux, and userland applications have begun to count on features we don't have as yet. A good example is the hardware context support in the i915 driver. This feature is required by Mesa 9.2, and Mesa 9.2 is itself required by the X.Org server 1.15. Furthermore, Radeon GPU support is way better in Mesa 10.x. Unfortunately, hardware context support was only added to the FreeBSD 10.1-RELEASE, meaning that we are stuck with Mesa 9.1 in older FreeBSD releases. Thus, our biggest, current challenge is to find an improved method of working with this DRM subsystem.



Using a Linux API Shim

A similar problem was solved in the Infiniband driver by embedding a Linux API shim (located in `sys/ofed/include/linux`). This layer either exposes a Linux API above FreeBSD native interfaces, or implements the Linux facilities. Thanks to this wrapper, the Infiniband driver remains close to the original code base targeted at Linux.

We could use a similar approach in the DRM subsystem by having our own shim. That's what was done by DragonFlyBSD and proved to be instrumental in bringing Intel Haswell GPU support in a very short time. Although I like this approach, I would prefer something centralized and shared among drivers, and this would obviously lower the maintenance effort. However, the Linux API is known to be a moving target and the reasons are well documented. If we take that road, we will need to answer the question of what to do if multiple drivers depend on incompatible versions of the Linux API.

The central shim solution was previously attempted at the time the Infiniband driver was imported, but eventually abandoned. One of the suggested answers was to settle on one popular Linux version for major distributions and stick to that in the shim. Besides the technical issue of the unstable API, most concerns were about the good or the bad this would do to FreeBSD regarding companies who do or do not provide FreeBSD native drivers.

In the case of DRM, I believe it would be a good thing and perhaps mandatory. During the XDC, I learned many facts that reinforced my opinion:

- AMD employs dozens of developers to work on their open-source driver. Double that for the Intel crew.
- AMD is planning to unify their closed-source Catalyst and open-source drivers to produce the open-source “amdgpu” driver, which will support newer hardware only. The current free driver will still be maintained to support older hardware.
- NVIDIA wants to change its proprietary driver so it registers as a DRM driver. It will remain a closed-source driver a user has to install from the Ports.
- Nouveau, the open-source driver for NVIDIA GPU, should be easy, though time-consuming, to port to FreeBSD because we have the TTM memory manager working. This would allow us to support those GPUs out-of-the-box, though performance wouldn't match the pro-

prietary driver. Furthermore, this would permit GPGPU support because NVIDIA doesn't provide a `libOpenCL.so` library for FreeBSD.

- ARM GPU drivers are mainly GPL today, but developers are willing to revise that to adopt a dual license model, like other drivers. The original choice of the GPL was mostly a “default” choice.

You can see that we could gain a lot by making the porting effort easier. I'm working on a new proposal which will be posted to the freebsd-arch@FreeBSD.org mailing list once ready.

DRM PRIME

After solving the maintenance issue with DRM, another important milestone is PRIME. DRM PRIME is an API exposed to userland applications to allow them to move data buffers between GPUs. In Linux, it is based on their dma-buf kernel facility.

When do we need this feature? First, it is required to support computers—most often laptops—equipped with two GPUs: one is a low-power GPU for basic use, the other is more powerful and targeted at more demanding tasks. In this case, the powerful GPU has no output connector: it is only a rendering engine and the first GPU has all the output connectors. Thanks to DRM PRIME, an application can use the second GPU to render a 3D image, take the resulting buffer and give it to the first GPU to display it on the screen.

Another use would be in a virtualized environment. A VM could use a physical GPU to render an image, get the result back, and use it in the virtual environment.

Mesa

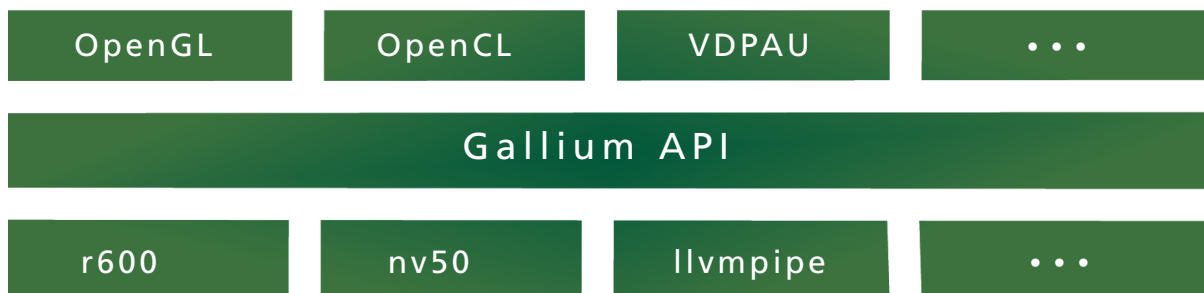
Issues with Mesa are smaller, but they are numerous. To better evaluate the importance of those problems, you have to first understand what Mesa is.

The Swiss Army Knife

Originally, Mesa was an open-source implementation of OpenGL. Today, it is one of the most critical userland components of the graphics stack. It offers:

- OpenGL and OpenGL ES APIs with either software or hardware rendering,
- hardware-assisted video decoding and encoding,
- an OpenCL implementation for GPGPU.

Mesa has become mandatory in many modern desktop environments. For example, compositors, those window managers who render and assemble application windows off-screen



and display the resulting image, take advantage of Mesa. If hardware rendering is unavailable, Mesa provides a software renderer based on LLVM powerful enough for this task.

Another example is Glamor: it provides 2D acceleration on top of OpenGL. Native hardware 2D acceleration is already available for many GPU, and several APIs have also been developed through the years, such as XAA, EXA, UXA or SNA (UXA and SNA being dedicated to Intel GPUs). These methods achieve very good 2D performance, at the cost of major efforts to maintain them. But 3D engines are powerful nowadays and X developers decided to use them for 2D acceleration as well—when a native method requires too much work, hence Glamor. Intel invests a lot of money in SNA and they will continue to use their native solution. On the other hand, AMD switched to Glamor as the 2D engine for their latest GPUs.

Similarly for other vendors—where the number of people working on their GPUs is limited—they can concentrate on the 3D driver and get the 2D driver almost for free.

For low-power platforms such as HTPC or mobile devices, hardware-assisted video decoding is very useful. Mesa supports this feature and makes it available through VDPAU, an API pushed by NVIDIA. The Intel flavor, VAAPI, may also be added in the future.

Beyond the desktop environments, Mesa includes Clover, a free implementation of OpenCL. This, however, is a work-in-progress and Clover is disabled by default in Mesa. Nonetheless, it is an important component for FreeBSD because vendors currently do not provide OpenCL libraries.

Gallium

In order to make all these tools and APIs avail-

RootBSD

Premier VPS Hosting

RootBSD has multiple datacenter locations,
and offers friendly, knowledgeable support staff.
Starting at just \$20/mo you are granted access to the latest
FreeBSD, full Root Access, and Private Cloud options.



www.rootbsd.net

able, Mesa has an internal infrastructure called Gallium.

Gallium is a low-level abstraction layer exposing the GPU functions. Above this layer Gallium state-trackers for each higher-level API (OpenGL, OpenCL, VDPAU, DirectX, etc.) transform the incoming call into Gallium calls. The benefit is to develop the Gallium driver for a GPU once and make it available through all high-level APIs.

udev

Except for the kernel driver, Mesa has little need from userspace. The only Linux-specific, required library is `libudev`. This library provides functions to list available devices, to get or set various attributes attached to them, and listen for incoming events. The implementation is mainly based on the `/sys` pseudo filesystem.

On FreeBSD, information can be obtained from `sysctl`s. The data queried by Mesa, especially Gallium, are:

- the vendor and device PCI IDs,
- the path of the device in `/dev`,
- the name of the kernel driver.

The file descriptor of an open device is passed to `udev` to get that. However, we are missing a link between this file descriptor and that information in FreeBSD.

For the time being, we have implemented a small library called `libdevq`, which answers Mesa-specific needs, but we would like it to be more generic. For example, to get the name of the driver and the PCI IDs, we need to query `"hw.dri.0.name"`, which is DRM-specific. Therefore, there is a need to augment kernel APIs to record the relation between a device driver instance and its entry in `/dev` and a `sysctl` tree (probably the `dev.* tree`).

We may want to extend `libdevq` to expose a `udev` API too. This would help with porting Mesa and other libraries such as `libinput`.

Piglit

Mesa developers publish an extensive test suite for Mesa called Piglit. It's an amazing tool for evaluating a new version of Mesa or updates to kernel drivers. However, we don't run it on a regular basis and we have to make `Piglit` easier to run on FreeBSD.

All dependencies are committed to the Ports tree. I think we are lacking a meta-port for installing Piglit dependencies. A user would have to install this port, then checkout and

build Piglit. Maybe a port to Piglit would be easier for the end-user, but the maintenance of such a port would require frequent updates. Piglit has not been released and the source repository receives new commits daily (improved tests and new tests). We need to use it each time we work on Mesa or the kernel. When we're confident with it, we should add this tool to the list of things people need to run when reporting bugs or when responding to a "Call for Testers."

Working with Upstream Developers

Mesa still needs some patches in order to build on FreeBSD, several to indicate that FreeBSD supports the same features as Linux. We have already tried to send them upstream to integrate them with Mesa. Unfortunately, they were lost in the mailing list, due to the large amount of mail and patches floating around, and our failure to call attention to them.

We need to better communicate with upstream developers to integrate FreeBSD patches when that makes sense. Thanks to the XDC, we were able to meet with people face to face, we talked about this and they are going to pay more attention to us. When we are comfortable with their workflow, they are even open to granting us commit privileges if we request them.

The FreeBSD Graphics Team

To read about the status of these projects or contact us, please visit our [wiki portal](#) or our brand new [blog](#)! •

Jean-Sébastien Pédrón has worked on the FreeBSD graphics stack since January 2013. He developed an interest in this area because his laptop's Radeon HD 5870 was barely supported. Now he spends a lot of

his spare time with the graphics team to improve the graphics stack on FreeBSD and advocate for it. When not working on FreeBSD, he plays drums and percussion and likes roller-skating, particularly at the [24 Hours of Le Mans](#).



FreeNAS

in an Enterprise Environment

By the time you're reading this, FreeNAS has been downloaded more than 5.5 million times. For home users, it's become an indispensable part of their daily lives, akin to the DVR. Meanwhile, all over the world, thousands of businesses, universities, and government departments use FreeNAS to build effective storage solutions in myriad applications.



What you will learn...

- How TrueNAS builds off the strong points of the FreeBSD and FreeNAS operating systems
- How TrueNAS meets modern storage challenges for enterprise

The FreeNAS operating system is free, open source, and offers thorough documentation, an active community, and a feature-rich storage environment. Based on FreeBSD, it can share over a host of protocols (SMB, NFS, FTP, iSCSI, etc) and features an intuitive web interface, the ZFS file system, a plug-in system for backup, and much more.

Despite the massive popularity of FreeNAS, many aren't aware of its big brother dutifully protecting data in some of the most demanding environments: the proven, enterprise-grade, professionally-supported line of TrueNAS.

But what makes TrueNAS different? Well, I'm glad you asked...

Commercial Grade Support

When a mission critical storage system fails, an organization's whole operation can come to a halt. Whole community-based support (like FreeNAS), it can't always get an answer and running in a timely manner. TrueNAS provides the responsiveness and expert support that a dedicated support team can provide that safety.

Created by the same team that developed FreeNAS.

WE INTERRUPT THIS MAGAZINE TO BRING YOU THIS IMPORTANT ANNOUNCEMENT:

THE PEOPLE WHO DEVELOP FREENAS, THE WORLD'S MOST POPULAR STORAGE OS, HAVE JUST REVAMPED TRUENAS.



POWER WITHOUT CONTROL MEANS NOTHING. TRUENAS STORAGE GIVES YOU BOTH.

- | | |
|---|--|
| <input checked="" type="checkbox"/> Simple Management | <input checked="" type="checkbox"/> Self-Healing Filesystem |
| <input checked="" type="checkbox"/> Hybrid Flash Acceleration | <input checked="" type="checkbox"/> High Availability |
| <input checked="" type="checkbox"/> Intelligent Compression | <input checked="" type="checkbox"/> Qualified for VMware and HyperV |
| <input checked="" type="checkbox"/> All Features Provided Up Front (no hidden licensing fees) | <input checked="" type="checkbox"/> Works Great With Citrix XenServer® |

To learn more, visit: www.iXsystems.com/truenas




POWERED BY INTEL® XEON® PROCESSORS

Intel, the Intel logo, Intel Xeon and Intel Xeon Inside are trademarks of Intel Corporation in the U.S. and/or other countries.


VMware and VMware Ready are registered trademarks or trademarks of VMware, Inc. in the United States and other jurisdictions.

Citrix makes and you receive no representations or warranties of any kind with respect to the third party products, its functionality, the test(s) or the results therefrom, whether expressed, implied, statutory or otherwise, including without limitation those of fitness for a particular purpose, merchantability, non-infringement or title. To the extent permitted by applicable law. In no event shall Citrix be liable for any damages of any kind whatsoever arising out of your use of the third party product, whether direct, indirect, special, consequential, incidental, multiple, punitive or other damages.

A high-angle, close-up photograph of a yellow excavator's arm and bucket. The arm is positioned diagonally from the top right towards the bottom left, digging into a large pile of dark, loose soil and rocks. The excavator's arm is bright yellow, contrasting with the dark, textured ground. The bucket is partially visible at the bottom right, having just dug into the earth. The ground is composed of dark, clumpy soil and numerous small, light-colored rocks and pebbles. The lighting is bright, casting shadows that emphasize the texture of the soil and the metallic surface of the excavator.

No More Printfs:

Digging into the Kernel with DTrace



By Mark Johnston



DTrace is a general-purpose performance analysis and tracing tool originally developed at Sun Microsystems for Solaris and later ported to a number of other operating systems. The last several major releases of FreeBSD have included a DTrace implementation, and DTrace support in the kernel has been enabled in **GENERIC** starting with FreeBSD 9.2 and 10.0.

DTrace is an enormously useful tool for gaining visibility into the inner workings of the FreeBSD kernel; its dynamic nature gives developers and administrators the ability to answer questions without the time-consuming kernel recompile and reboot required by many other kernel debugging tools. As such, DTrace belongs in every kernel developer's toolbox. As with any tool, however, DTrace has its share of limitations and problems; it is important to be aware of them and know how to work around them.

This article will illustrate a few choice DTrace tricks and techniques that one can use to help answer questions about the kernel's activity. One commonly-cited capability of DTrace is the ability to instrument the entry and return of arbitrary kernel functions in real time, as well as print their arguments and return values. Though this is in itself quite useful, it is often insufficient when attempting to track or isolate complex behavior within the kernel: some functions (`tcp_do_segment()`, for example) are extremely long, so function-level tracing is of limited use. One may also wish to keep track of state between probes, for example when tracing memory allocations. Through examples, we will develop some more sophisticated techniques for using DTrace to answer questions and troubleshoot problems. Additionally, some common mistakes and gotchas will be highlighted.

We'll begin with a quick refresher on DTrace for those who have encountered it before, but those seeking a comprehensive introduction are strongly encouraged to read "*The DTrace Book*" by Brendan Gregg and Jim Mauro[1]. The rest of the article assumes a basic grasp of DTrace and FreeBSD kernel internals.

An Ultra-Quick Overview

DTrace itself is made up of several components:

- `dtrace(1)`, the main command-line interface to DTrace, is used to execute DTrace scripts, query the kernel for information about DTrace probes, and perform several build-time functions. FreeBSD includes a number of other more specialized DTrace-based tools, including `lockstat(1)` and `dtruss(1)`.
- The D programming language, used to write DTrace programs which are compiled into bytecode by `libdtrace` and handed to the kernel for execution.
- DTrace probes, which are the points of instrumentation for DTrace. A D script can be used to enable one or more probes within the kernel or a userland program. When an enabled probe fires,

the script determines what action, if any, to take. A full listing of available probes can be viewed by running `dtrace -l` on a FreeBSD system. DTrace providers are collections of related probes. For example, the FBT (function boundary tracing) provider defines a probe at the beginning of every kernel function; such probes are called `fbt::<function_name>:entry`. The IP provider defines a small set of probes that correspond to events in FreeBSD's IPv4/IPv6 stacks.

A D program consists of a list of probe-predicate-action tuples—each tuple specifies one or more probes, the action(s) to be taken if one of the probes fires, and an optional predicate that dynamically determines whether the actions are to be taken when the probe fires. Syntactically, each tuple resembles an AWK program:

```
probe
/predicate/
{
    actions
}
```

For example, here is a D script which prints the source address of every IPv6 packet received on the `re1` interface:

```
ip:::receive
/args[2]->ip_ver == 6 && args[3]->if_name == "re1"/
{
    printf("%s", args[2]->ip_saddr);
}
```

In this example, the `ip:::receive` probe fires whenever an IP packet is received by the kernel, the source address is stored in `args[2]->ip_saddr` and printed using `printf()`, and the `printf()` is only executed if the packet's IP version is 6 and the receiving interface is named `re1`. The script can be run by copying it to a file (usually given a `.d` extension) and executing `dtrace -s script.d`.

The `ip:::receive` probe is an example of a statically-defined probe—the probe point is embedded at one or more locations within the kernel, and has little to no overhead when disabled. DTrace also creates a number of probes on the fly when it is first loaded—these are dynamic probes. The main example of these is the FBT provider, which creates `entry` and `return` probes for each C function in the kernel. These can be used to obtain a fine-grained view of the kernel's behavior; however, as the details of kernel internals change from one release to the next, D scripts written for one kernel may not work with another. Statically-defined probes can provide more stability since they correspond to high-level logical events rather than being tied to implementation details.

Tracking Down Memory Leaks

Dynamic probes, despite their shortcomings, tend to be very useful in narrowing down the source of a problem. D scripts written as part of an investigation are usually just one-off tools to help isolate a specific behavior, so the lack of probe stability is not an issue. `dtrace(1)`'s output is also quite customizable, so it is often convenient and useful to combine DTrace with other programs. Post-processing the raw output of a D script is a common technique used, for example, to create Flame Graphs[2].

Finding the cause of a memory leak can be an arduous process. This is especially true in the kernel, which lacks the sophisticated debugging tools available for userland programs, and where the `malloc()` interface has a large number of disparate users. FreeBSD's DTrace implementation has a `dtmalloc` provider which can help quickly track down the source of a memory leak. Before introducing this provider however, let us revisit the `malloc()` interface provided in the FreeBSD kernel:

```
void *malloc(unsigned long size, struct malloc_type *type, int flags);

void free(void *addr, struct malloc_type *type);
```

In particular, the kernel interface includes a `malloc_type` argument, which is used to distinguish

among various subsystems and track allocation statistics on a per-subsystem basis. The various malloc types and their usage can be viewed by running `vmstat -m`.

The `dtmalloc` DTrace provider defines probes for memory allocation events and ties into the malloc types by defining probes for each type. These probes can be viewed by running `dtrace -l -P dtmalloc`. We can get a quick picture of the most active subsystems with the following one-liner:

```
# dtrace -n 'dtmalloc:::malloc {@[probefunc] = count()}'
```

This program will print the names of the active malloc types sorted by allocation count when it is ended with a Ctrl-C:

```
...
jnewblk          732
newblk           732
temp             918
jsegdep          1381
rpc              3358
iov              3443
NFS_fh           5817
```

Of particular note is the appearance of the `temp` type, which is used for short-lived allocations that are not tied to any particular kernel subsystem. A useful extension of this one-liner is to record and display the most active malloc types per second:

```
dtmalloc:::malloc
{
    @types[probefunc] = count();
}

tick-1s
{
    printa(@types);
    trunc(@types);
}
```

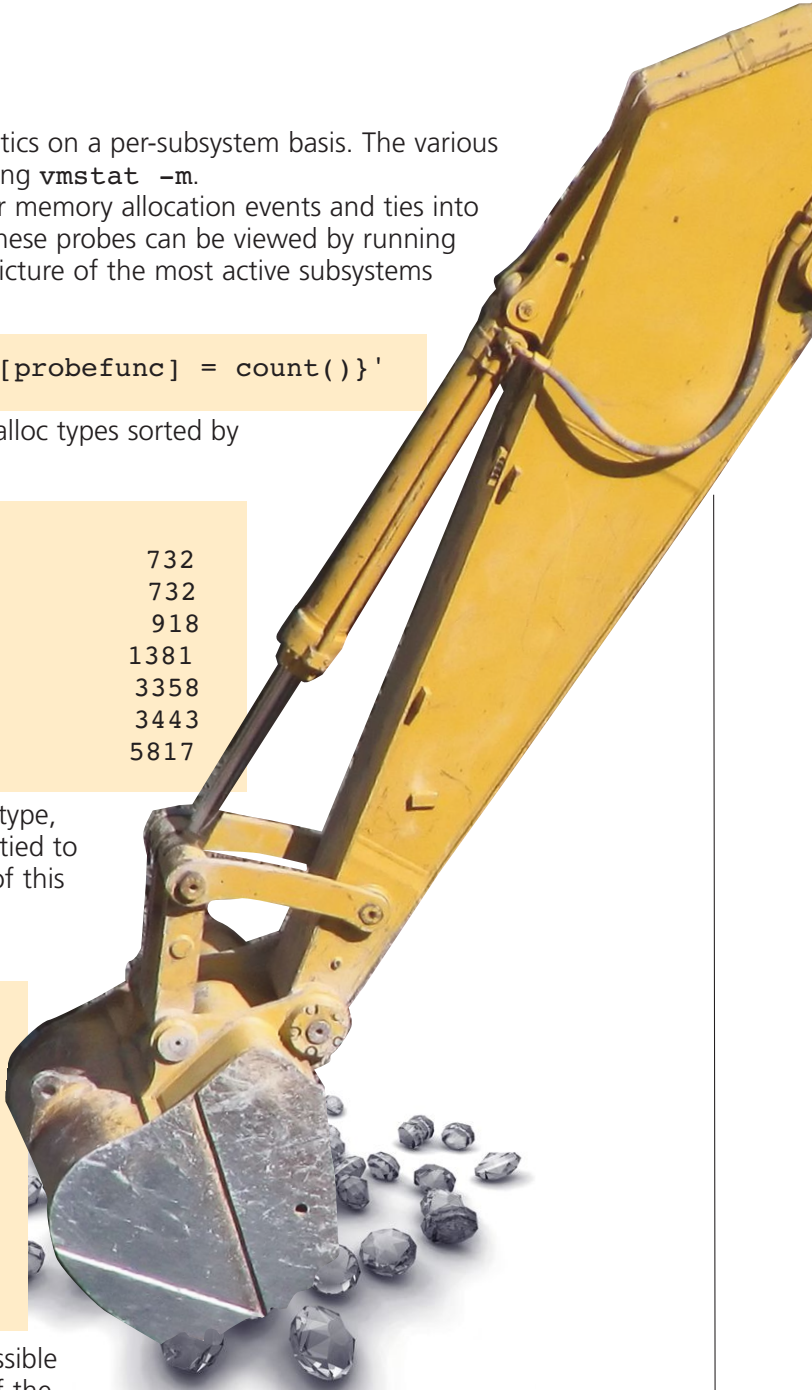
On sufficiently recent versions of FreeBSD, it is possible to use DTrace's new `aggpack` option to get a view of the size distribution of allocation requests on a per-subsystem level:

```
# dtrace -n 'dtmalloc:::malloc {@[probefunc] = quantize(args[3]);}' -x aggpack
```

If a subsystem is leaking memory allocated by `malloc()`, we can use the `dtmalloc` and `fbt` providers to help pinpoint the code path(s) responsible for the leak. To illustrate this, consider the following simple scenario: after running a test which makes use of `pmc(3)`, `hwpmc(4)` is observed (via a console message) to leak memory on unload.

Warning: memory type `pmc` leaked memory on destroy (8 allocations, 1024 bytes leaked).

The first step toward fixing this is to figure out when and how this memory is being allocated. The `hwpmc(4)` code contains roughly 45 calls to `malloc(9)`—trying to track this down with code inspection is unlikely to work, especially for one unfamiliar with the `hwpmc(4)` code. This is a good opportunity to use D's `stack()` function to identify the code path(s) responsible for this leak:



Digging into the Kernel

```
#pragma D option quiet

dtmalloc::$1:malloc
{
    self->trace = 1;
}

fbt::malloc:return, fbt::contigmalloc:return
/self->trace == 1/
{
    printf("alloc 0x%p", args[1]);
    stack();
    self->trace = 0;
}

fbt::free:entry, fbt::contigfree:entry
{
    self->addr = (uintptr_t)args[0];
}

dtmalloc::$1:free
/self->addr != 0/
{
    printf("free 0x%p\n", self->addr);
    self->addr = 0;
}

fbt::free:return, fbt::contigfree:return
/self->addr != 0/
{
    self->addr = 0;
}
```

When run with an argument of `pmc`, this script prints a stack trace for every allocation of `pmc` memory, as well as the address of the allocated memory; when `pmc` memory is freed, its address is printed. It makes use of thread-local variables (prefixed by `self->`) to keep track of state between successive probes. Indeed, when the `dtmalloc::$1:malloc` probe fires, a thread-local variable is set so that DTrace knows to print the stack on return from `malloc(9)`. Similarly, when freeing memory, we keep track of the freed address so that we can log it once it is confirmed to be of the correct type. It is important to set `self->addr` back to 0 on return from `free(9)`, since this ensures that DTrace releases the dynamic storage associated with the variable.

Note that this script does not instrument `realloc(9)`: under the hood, `realloc(9)` is implemented by allocating a new memory block using `malloc(9)`, copying the contents of the original block to the new one, and freeing the original. Therefore, if a `realloc(9)`-allocated block is leaked, we will have printed a stack trace at the time of the last `realloc(9)` call.

Running the script produces output which can then be post-processed with a quick Perl script:

```
my %stacks;
while (<>) {
    if (/^alloc (0x[a-f0-9]+)$/) {
        while (<>) {
            last if /^[af]/;
            stacks{$1} .= $_;
        }
    }
}
```

CODE CONTINUES NEXT PAGE


```

    }
    if (/^free (0x[a-f0-9]+)$/ ) {
        delete $stacks{$1};
    }
}

foreach my $key (keys %stacks) {
    print $stacks{$key} . "\n";
}

```

Running this script on the output of the D script yields the following output:

```

hwpmc.ko`pmc_syscall_handler+0x1fb6
kernel`amd64_syscall+0x25a
kernel`0xffffffff809342cb

hwpmc.ko`pmc_syscall_handler+0x1fb6
kernel`amd64_syscall+0x25a
kernel`0xffffffff809342cb

```

Thus, the memory leaks we saw must originate at `pmc_syscall_handler+0x1fb6`; the corresponding code can then easily be found using `kgdb(1)`:

```

# kgdb
...
(kgdb) list *pmc_syscall_handler+0x1fb6

```

This turns out to be in the `PMC_OP_PMCALLOCATE` handler, which allocates PMCs. Now that we've established where the leaked memory comes from, we have a much better shot at finding the root cause of the memory leak.

Before proceeding, it is worth examining the D script above in a bit more detail. A different approach might have been to track the memory (de)allocations in the D script itself, using an associative array indexed by allocation address. This approach can also be useful, but has some drawbacks:

- DTrace reserves a fixed amount of memory for global variables. If the subsystem(s) being traced have a large number of long-term allocations, DTrace may run out of memory to store them.
- DTrace doesn't provide an easy means to save a stack trace into a variable; the following is not permitted, for instance:

```

fbt::malloc:return, fbt::contigmalloc:return
/self->trace == 1/
{
    stacks[args[1]] = stack();
    self->trace = 0;
}

```

It is however possible to use the `stack()` function output as an aggregation key.

- Post-processing allows for more flexibility. If we were tracing a subsystem that always has some outstanding allocations, we would have to be careful to distinguish them from leaked memory. It would be straightforward to extend the post-processing approach to include a timestamp with each allocation; storing additional data in an associative array adds overhead, both in the time required to perform the probe action and the memory required to store it.

Building Associations

A common stumbling block when building monitoring tools with DTrace is the fact that the data one needs may be scattered among multiple probe sites. Consider the task of monitoring UDP traffic by process. A natural way to start would be to try using the `udp:::send` and `udp:::receive` probes to count bytes or packets. But as it turns out, on FreeBSD the arguments to these probes do not give

Digging into the Kernel

any way to discern the process responsible for the traffic. The `curthread` variable is not particularly helpful here either—a packet's path between a socket buffer and a network interface is generally handled by `netisr(4)` threads—dedicated interrupt-priority threads that handle protocol processing for network packets:

```
# dtrace -n 'udp:::receive {printf("%s", curthread->td_name);}'
dtrace: description 'udp:::receive ' matched 1 probe
CPU      ID                      FUNCTION:NAME
  1    37374                      :receive swil: netisr 0
  6    37374                      :receive swil: netisr 0
  5    37374                      :receive swil: netisr 0
  5    37374                      :receive swil: netisr 0
```

At this point, one is inclined to give up. There is indeed no direct way to find the associated process from the context of the "udp" provider probes. However, we can leverage the fact that the `cs_cid` field of the second argument to `udp:::send` and `udp:::receive` is a pointer to the `struct inpcb` (internet protocol control block) associated with the packet. This structure holds the connection state of TCP and UDP sockets, and in particular contains a pointer to the socket. Now, using FBT probes, we can perform an action anytime a process creates a UDP socket, and use an associative array to map the PCB to the PID of the process. Then in the UDP probes, we can use the PCB's address to look up the PID:

```
fbt::udp_attach:entry
{
    self->so = args[0];
}

/*
 * Wait until udp_attach() has returned successfully before entering anything
 * into our map - we don't want to do anything if there was an error.
 */
fbt::udp_attach:return
/args[1] == 0 && self->so != NULL/
{
    procs[(uintptr_t)self->so->so_pcb] = curproc->pr_pid;
    self->so = 0;
}

/*
 * Be sure to free array entries once the socket is no more.
 */
fbt::in_pcbdetach:entry
{
    procs[(uintptr_t)args[0]] = 0;
}
```

Then, in the UDP probes, we can use the PCB address to perform lookups in the `procs` array:

```
udp:::send, udp:::receive
/procs[args[1]->cs_cid] != 0/
{
    /* Do stuff with the PID. */
}
```

Of course, this script will not be able to record the PID for a socket if the socket has already been created by the time the script is run. In addition to updating the `procs` array in `udp_attach()` and `udp_detach()`, we

can opportunistically update the map whenever the process performs I/O on a UDP socket:

```
fbt::sosend_dgram:entry, fbt::soreceive:entry
/args[0]->so_proto->pr_protocol == IPPROTO_UDP/
{
    procs[(uintptr_t)args[0]->so_pcb] = curproc->pr_pid;
}
```

This lets our hypothetical monitoring tool handle long-lived connections just as easily as short DNS lookups.

This general trick of collecting information by building lookup tables is quite powerful, though it requires a certain amount of familiarity with the kernel and how its data structures relate to each other. Another application is in mapping file paths to vnodes. Vnodes are the FreeBSD kernel's in-memory representations of files and are used to cache various pieces of information about files as they are accessed. However, the file system paths used to look up files are stored in a separate cache, the name cache. In particular, paths are not stored together with vnodes, so given a vnode, there is no straightforward way to find an associated path from within a D script. However, we can hook into the name cache lookup functions and build up a mapping table keyed by vnode address:

```
fbt::vn_fullpath1:entry
{
    self->vn = args[1];
    self->buf = args[4];
}

fbt::vn_fullpath1:return
/args[1] == 0 && self->buf/
{
    paths[self->vn] = stringof(*self->buf);
    self->vn = 0;
    self->buf = 0;
}

vfs::lookup:hit
/paths[args[0]] != "" && paths[args[2]] == ""/
{
    paths[args[2]] = strjoin(paths[args[0]], strjoin("/",
stringof(args[1])));
}

fbt::cache_purge:entry
/paths[args[0]] != ""/
{
    paths[args[0]] = 0;
}
```

Looking at Process Argument Vectors

One classic DTrace script is `execsnop`, written by Brendan Gregg and available in the DTrace toolkit[3]. It allows users to watch process execution in real time by tracing the `curpsinfo->pr_psargs` variable (defined in `/usr/lib/dtrace/psinfo.d`) on return from the `execve(2)` system call; a simplified version is here:

```
#pragma D option quiet

syscall::execve:return
{
    printf("%s\n", curpsinfo->pr_psargs);
}
```


Digging into the Kernel

Running `execsnoop` (or this example) on a FreeBSD system may result in some intimidating-looking warnings:

```
...
cc --version
sh -c echo 3.4.1 | awk -F. '{print $1 * 10000 + $2 * 100 + $3;}'
awk -F. {print $1 * 10000 + $2 * 100 + $3;}
dtrace: error on enabled probe ID 1
      (ID 39505: syscall:freebsd:execve:return):
      invalid address (0x4) in action #1 at DIF offset 136
```

This turns out to be an artifact of some behavior in the FreeBSD kernel: the kernel caches process arguments in the `struct proc` associated with the process, but only if they fit in the limit defined by the `kern.ps_arg_cache_limit` sysctl. This sysctl has a default value of 256, which is insufficient for many programs. One solution would be to just increase the sysctl value as needed, but this is less than ideal—scripts that depend on a custom system configuration are harder to reuse, and modifying this configuration might have some unforeseen consequences.

However, there is one point at which the full argument vector is available in the kernel—upon entry into the `kern_execve()` function. Thus, a different approach is to copy the arguments when `execve(2)` first enters the kernel, and print them once DTrace has confirmed that the `execve(2)` system call was successful. This can be done using DTrace speculations:

```
#pragma D option nspec=32
#pragma D option quiet
#pragma D option strsize=4096

syscall::execve:entry
{
    self->argv = speculation();
}

fbt::kern_execve:entry
/self->argv/
{
    speculate(self->argv);
    printf("%s\n", memstr(args[1]->begin_argv, ' ',
        args[1]->begin_envv - args[1]->begin_argv));
}

syscall::execve:return
/args[1] == 0 && self->argv/
{
    commit(self->argv);
    self->argv = 0;
}

syscall::execve:return
/args[1] != 0 && self->argv/
{
    discard(self->argv);
    self->argv = 0;
}
```

There are several less commonly-used DTrace features at work in this script. First and foremost is the `self->argv` variable, which is used in combination with a number of D functions. This is a use of

DTrace's speculative tracing feature, which allows us to capture data before we know whether we want it. Here we use the arguments to `kern_execve()` to extract the arguments of a program, but it would not be correct to print them in the `fbt::kern_execve:entry` probe, since the system call may still fail. The solution is to speculatively trace the arguments; once we know that the system call has succeeded, they can be printed. Otherwise, the string is discarded.

Speculations involve four different D functions. First, `speculation()` is used to allocate a speculation buffer. A handle for this buffer is stored in the `self->addr` thread-local variable, which is passed as an argument to the other speculation functions. The `speculate()` function enables speculative tracing for the remainder of the probe—the output of data-gathering actions is stored in the speculative buffer for later use. Finally, the `commit()` and `discard()` functions respectively store the speculative data in DTrace's output buffers, and discard it. DTrace can only allocate a fixed number of speculation buffers; the `nspec` option (used in the example script above) can be used to tune the number of buffers available. Comprehensive documentation for DTrace's speculative tracing feature is available at [4].

The other unusual aspect of this example is the use of the `memstr()` D function. At the time of writing, this function is unique to FreeBSD and was specifically added to handle FreeBSD's memory layout for argument strings in the kernel. For example, the string `wc -w article.txt` would be stored as

w	c	\	-	w	\	a	r	t	i	c	e	.	t	x	t	\
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

where `\` denotes the null byte. As in C, D strings are null-terminated, so a `printf()` of the above string would only return `wc`. Because D gives us no way to iterate over each component of the string, the `memstr()` function was added to convert its first argument to a D string by replacing all occurrences of the null byte with its second argument. The third argument indicates the length of the first argument.

Conclusion

We hope that the examples above serve as a small testament to DTrace's adaptability. As with any introspection tool, there are limits to DTrace's abilities; the ability to trace kernel execution and subroutine arguments at the function call level is a powerful feature indeed, but it is often insufficient on its own when peering into a running kernel and asking questions about its behavior. Moreover, the ability to trace in arbitrary kernel contexts imposes many restrictions on what we are allowed to do in a probe action. On the other hand, DTrace gives us many features to solve, or at least work around, these limitations, and the ongoing development of DTrace in both FreeBSD and illumos continues to help expand its utility. •

Mark Johnston is a software engineer living in Seattle. Originally from Toronto, he completed a degree in mathematics at the University of Waterloo in 2013 and has been a FreeBSD user since 2010. Since obtaining a commit bit, his main focus has been on improving FreeBSD's DTrace implementation. He can be reached via email at markj@FreeBSD.org.

REFERENCES

- [1] *DTrace: Dynamic Tracing in Oracle Solaris, Mac OS X, and FreeBSD* by Brendan Gregg and Jim Mauro, Prentice Hall 2011.
- [2] CPU Flame Graphs, <http://www.brendangregg.com/FlameGraphs/cpuflamegraphs.html>
- [3] DTrace Toolkit, <http://www.brendangregg.com/dtracetoolkit.html>
- [4] Speculative Tracing, <https://wikis.oracle.com/display/DTrace/Speculative+Tracing>





pfSense

A 3-HOUR TOUR

FreeBSD in combination with a variety of its ports can make for a powerful firewall solution comparable to, and sometimes better than, commercial offerings. One potential downside to rolling your own is the learning curve involved in manually configuring all the various underlying components. The required knowledge base makes building and maintaining such a solution inaccessible to much of the market. There is also a requirement for a significant amount of “glue code” between various pieces to enable the whole to work in a cohesive manner. pfSense® software has been filling this gap, for over a decade. pfSense software makes FreeBSD firewalls accessible to every IT professional capable of managing a typical commercial-grade firewall. With pfSense software, everything is managed via an easy-to-use web interface that's similar to the GUIs of Cisco ASAs, Watchguard, Sonicwall, and other products. The pfSense project is to network security and FreeBSD what FreeNAS is to Network Attached Storage and FreeBSD. Over the past decade, we've grown to 300,000 known live installs.

While our user base initially comprised mostly users who had no, or very little, knowledge of how to manually configure anything in the underlying components, that's changed significantly in the last few years. When the project first started, we often had to educate new users that FreeBSD is not Linux—in fact, it's better. As we grew, some highly-skilled BSD sysadmins scoffed at the idea of using a GUI to manage a firewall. Many of these experts have since gained an understanding of the value and time savings in using pfSense software.

Understand that the glue tying various components together is beyond new users of BSD firewalls, and can be daunting even for the most seasoned professional. As a common example, consider the scenario of an Internet connection with a dynamic IP Internet address.

By
Chris Buechler
and
Jim Thompson

Depending on the subsystems one has configured, when an IP changes, there are several things that may need to occur, including: dynamic DNS updates, clearing cached firewall and NAT states from the previous IP; configuration files for select services need to be updated with the new IP, and any affected services subsequently restarted. Stock BSD systems don't have easy facilities to handle circumstances like this, much less to do it all automatically. Even seasoned sysadmins appreciate having firewalls that more junior staff can easily manage. Eliminating 3 a.m. trouble calls can make for a less grumpy sysadmin.

pfSense software release 2.2 is the culmination of 15 months of development effort in a number of areas. This article highlights the most significant changes in release 2.2.

Base OS Upgrade

In the open-source model it is typical that distributions (downstream) tend to lag the individual software components (upstream). Historically, pfSense has been subject to the same dynamic. pfSense software version 2.1 was released September 15, 2013. 2.1 and its subsequent releases were based on FreeBSD 8.3, which was released in April 2012. While we backported some newer drivers, improving hardware support compared to stock FreeBSD 8.3, the result is that FreeBSD 8.3 was no longer officially supported on April 30, 2014, well before pfSense version 2.2 was released.

Seeking to improve this situation, during development of this release, we tracked FreeBSD 10-STABLE and then 10.1 from BETA to RC to RELEASE. With release of pfSense version 2.2, we've caught up to the most recent FreeBSD release, and are working to ease and simplify the process of changing base OS versions in future releases. Expect to see us keeping much more up-to-date with the latest and greatest in FreeBSD in the future.

FreeBSD 10.1 brings with it several technical improvements. One of the most interesting to our community is the improvements to pf. FreeBSD developer Gleb Smirnoff made pf SMP-friendly for FreeBSD 10. pf now supports fine-grain locking and better utilization of CPUs on multi-core machines. While we're happy to leverage the benefits of open source, we're also happy to contribute, so we made pf even faster. We started an investigation with FreeBSD developer George Neville-Neil into further improving

FreeBSD

VPS HOSTING in the Cloud

- Deploy in 30 Seconds
- 24/7/365 Live Support
- No Contract, Cancel Anytime
- Free Setup, Free DNS
- 100% Enterprise SSD
- 1TB Outbound Transfer
- 1 Gbps Port

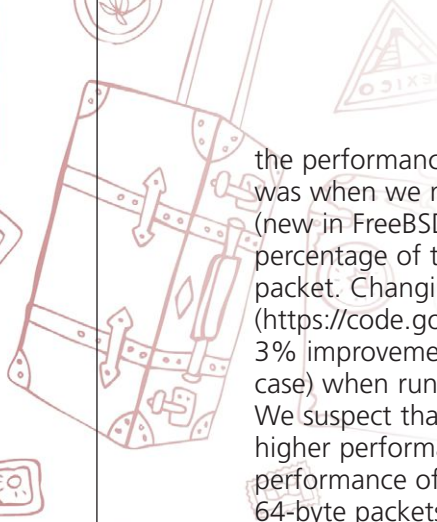
Plans start at only

99¢
per month

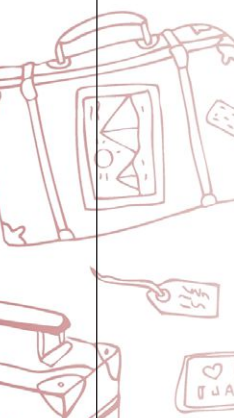


atlantic.net

GET STARTED TODAY!



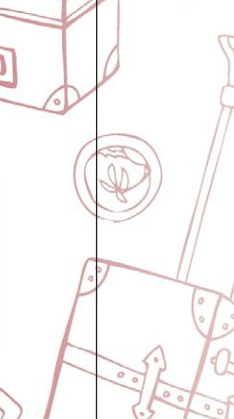
the performance of pf. One of the first results was when we noticed that the Jenkins hash (new in FreeBSD 10) was consuming a sizable percentage of the time spent processing each packet. Changing the Jenkins hash to xxhash (<https://code.google.com/p/xxhash/>) results in a 3% improvement in the null pf test case (worst case) when run over Intel ixgbe(4) interfaces. We suspect that other hardware will yield even higher performance gains, as raw forwarding performance of ixgbe(4) is only about 550,000 64-byte packets per second on the hardware used for testing.



Improved virtualization support as a guest in FreeBSD 10.1 is another thing our users have been taking advantage of for months in our 2.2 beta and release candidate snapshots. While this has worked very well in previous releases for years with most hypervisors in most circumstances, 10.1 brings the first really solid solution for Microsoft's Hyper-V, and a few have reported virtio improvements.

The improved 10-Gb network drivers in 10.1 are another welcome enhancement for our users.

IPsec Enhancements



The Internet Protocol Security (IPsec) suite is used to implement virtual private networks on FreeBSD and pfSense software. Traditionally, system and network administrators were required to make a choice between desired security levels and the performance requirements in the network. As the networking world continues its transition from 1 to 10, to 40 gigabits per second speeds and faster, improvements in IPsec's cryptographic building blocks must also keep pace.

The FreeBSD Foundation and Netgate worked jointly to have FreeBSD developer John-Mark Gurney add AES-CTR and AES-GCM modes, including acceleration using Intel's AES-NI instructions to FreeBSD's cryptographic framework. AES-GCM is an authenticated encryption algorithm, ideal for protecting packetized data, because it has minimum latency and minimum operation overhead.

Concurrent with this project, FreeBSD developer and pfSense software developer Ermal Luci updated the FreeBSD IPsec stack to support RFC 4106 and RFC 4543, which standardizes the use of GCM in IPsec Encapsulating Security Payload (ESP), and Galois Message Authentication Code in IPsec ESP and AH.

The back-end keying daemon for IPsec has changed from ipsec-tools (racoon) to

strongSwan. The primary reason for this change is to include support for IKEv2, while simultaneously supporting IKEv1. While we considered OpenIKED from OpenBSD, there wasn't a straightforward way to simultaneously support IKEv1 for those who need it. StrongSwan supports IKEv1 and fully implements IKEv2. The IKEv2 daemon from strongSwan is multi-threaded (16 threads by default). Using strongSwan, it has been shown that up to 20,000 concurrent IPsec tunnels can be handled on suitable hardware when used as a VPN gateway. StrongSwan supports Elliptic Curve Cryptography (ECDH groups and ECDSA certificates and signatures) both for IKEv1 and IKEv2, providing interoperability with Microsoft's Suite B implementation on Windows Vista, 7, 8, Server 2008, 2012 and newer.

An additional benefit of the change to strongSwan is more flexible debug logging configurations. With ipsec-tools, we had two options—log normally, or enable debug logging. With strongSwan, there are 16 different areas where the logging levels can be individually configured. This can help generate more useful logs in troubleshooting circumstances, without being inundated with log noise of cranking up debug logging everywhere.

Layer Two Tunneling Protocol (L2TP) is an industry standard tunneling protocol that provides encapsulation for sending Point-to-Point Protocol (PPP) frames across packet-oriented media. Since L2TP does not provide any encryption or authentication, it is typically combined with IPsec. With pfSense software 2.2, L2TP/IPsec is now supported, enabling easy transport of non-IP protocols inside a secure VPN.

With IKEv2 and L2TP/IPsec, we've added two new VPN options with clients built into Windows Vista and newer, as well as many other operating systems. Let's hope this finally kills off PPTP, for which we've had a big red security warning for the last several versions strongly encouraging people switch to a VPN that's actually secure. The last of the "it's easy because it's built into Windows" argument is gone.

DNS Resolver Changes

Having a local DNS resolver on the firewall is helpful in networks without a local DNS server, as it offers a local DNS cache, as well as the ability to configure local host and domain overrides for name resolution. pfSense software also provide the ability to automatically register DHCP leases and their associated hostnames in

the DNS forwarder, so for the most basic of networks, it can serve all the local DNS needs, including internal name resolution. Our 2.1.x and prior releases used Dnsmasq as the caching DNS forwarder, configured under Services>DNS Forwarder. Dnsmasq came enabled by default, using the system's configured DNS servers to perform its lookups. Dnsmasq is strictly a forwarder that relies on recursive DNS servers to perform its lookups. Where there are no internal DNS servers, it requires using a DNS server on the Internet such as Google public DNS or OpenDNS, or using your ISP's DNS servers. Unbound is capable of recursion and comes with it enabled by default, so it isn't dependent on a specific list of DNS servers to function.

Following in the footsteps of FreeBSD 10, pfSense version 2.2 changes the default DNS resolver to Unbound for new installations. Systems upgraded from a prior release and existing configurations from earlier versions restored onto 2.2 will retain the previous behavior, using Dnsmasq as the DNS forwarder. The benefits of Unbound as a DNS resolver include improved scalability, improved performance with large cache sizes, DNSSEC support, and the ability to perform recursive queries directly on the pfSense platform, rather than relying on other, external DNS servers. Unbound is configured under Services>DNS Resolver.

High Availability

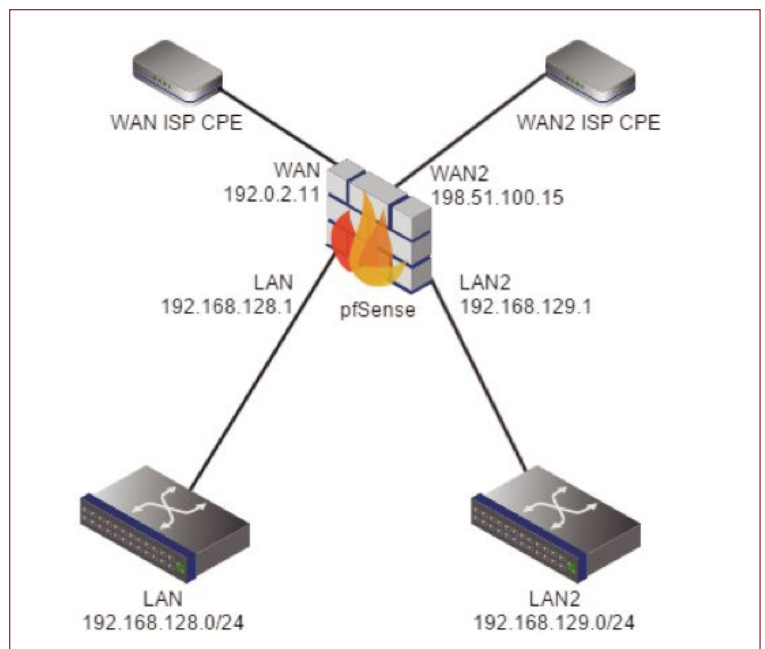
The combination of CARP, pfsync, and our XML-RPC-based configuration sync provides high availability functionality in pfSense software. This ensures seamless failover in the event of a hardware failure, or failure of network connectivity between primary and secondary for any reason. High availability also means easier, uninterrupted maintenance. By leveraging the high availability components, a firewall cluster can be upgraded node-by-node without service disruptions. To assist with this, pfSense software version 2.2 adds a new feature called maintenance mode. Maintenance mode works by bumping the advertising skew value in CARP to prevent a system from taking master status unless the backup system is no longer reachable. This is helpful in circumstances where a requirement exists to keep the primary system as backup for some reason, typically because of a hardware problem or other maintenance issues with the primary, where a need to confirm a problem is fixed before putting the system back into service exists.

Outbound NAT Enhancements

Outbound NAT translates the source IP and port on traffic leaving a specific interface. Most commonly this is to translate internal RFC 1918 private IP space to public IPs. Earlier releases of pfSense software offered two options for outbound NAT—automatic and manual.

For example, consider the following network with two Internet connections (WAN and WAN2) and two internal networks (LAN and LAN2) (Figure 1).

For this network, the default automatic outbound NAT works as follows. Traffic leaving WAN sourced from either LAN or LAN2 will have its source IP translated to the WAN interface IP. Traffic leaving WAN2 sourced from LAN or LAN2 will have its source IP translated to WAN2's IP. By default, pfSense rewrites the source port on all NATed traffic except ISAKMP (UDP port 500). This is the desired behavior in most environments that use a single public IP per WAN. Traffic between internal networks does not have



NAT applied.

For modes where automatic outbound NAT rules are in use, there is now a list of the automatic rules shown on the outbound NAT configuration screen under any user-defined outbound NAT rules (Figure 2).

In cases where multiple public IPs exist per WAN, or a requirement to not rewrite the source port on some of the traffic exists, the situation is more complicated. For these circumstances, outbound NAT can be changed to man-



Automatic rules:

Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description
▶ WAN	127.0.0.0/8 192.168.128.0/24 192.168.129.0/24	*	*	500	192.0.2.11	*	YES	Auto created rule for ISAKMP
▶ WAN	127.0.0.0/8 192.168.128.0/24 192.168.129.0/24	*	*	*	192.0.2.11	*	NO	Auto created rule
▶ WAN2	127.0.0.0/8 192.168.128.0/24 192.168.129.0/24	*	*	500	198.51.100.15	*	YES	Auto created rule for ISAKMP
▶ WAN2	127.0.0.0/8 192.168.128.0/24 192.168.129.0/24	*	*	*	198.51.100.15	*	NO	Auto created rule

FIG. 2

ual mode, where one must manually configure all outbound NAT rules. Upon initially changing from automatic to manual outbound NAT, the complete outbound NAT ruleset that was being automatically generated is added. However, this list is static, and must be manually maintained thereafter.

Snipped from the bottom of Figure 3 are the entries for the LAN2 192.168.129.0/24 subnet, which aside from source network are identical to the entries for LAN.

Consider the situation where a third LAN

network is added (LAN3) with a network of 192.168.130.0/24, and rules are in place permitting that network to reach the Internet. Because LAN3's subnet has no matching outbound NAT rules, the traffic is passed out the WAN interface with an RFC1918 source address, and the connection will fail.

One solution would be to duplicate the existing LAN or LAN2 outbound NAT rules for each WAN, but a better solution is to relax mask on the existing outbound NAT rule to cover all of the existing internal RFC1918 numbered networks.

Mappings:

Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description
▶ WAN	127.0.0.0/8	*	*	500	WAN address	*	YES	Auto created rule for ISAKMP - localhost to WAN
▶ WAN	127.0.0.0/8	*	*	*	WAN address	*	NO	Auto created rule - localhost to WAN
▶ WAN2	127.0.0.0/8	*	*	500	WAN2 address	*	YES	Auto created rule for ISAKMP - localhost to WAN2
▶ WAN2	127.0.0.0/8	*	*	*	WAN2 address	*	NO	Auto created rule - localhost to WAN2
▶ WAN	192.168.128.0/24	*	*	500	WAN address	*	YES	Auto created rule for ISAKMP - LAN to WAN
▶ WAN	192.168.128.0/24	*	*	*	WAN address	*	NO	Auto created rule - LAN to WAN
▶ WAN2	192.168.128.0/24	*	*	500	WAN2 address	*	YES	Auto created rule for ISAKMP - LAN to WAN2
▶ WAN2	192.168.128.0/24	*	*	*	WAN2 address	*	NO	Auto created rule - LAN to WAN2

FIG. 3

Due to the unlikelihood of having internal clients using IPsec without NAT-T, in most cases the static port rule for ISAKMP may also be safely removed. Most systems also don't need the 127.0.0.0/8 as a source. This is there for circumstances where binding a service specifically to localhost is required. Thus the original 12 outbound NAT rules can be narrowed down to just two, as shown in Figure 4.

Still, these are manually maintained rules. If an internal network outside of 192.168.0.0/16, or a new Internet connection is added, a manual update of the outbound NAT rules will still be required.

pfSense version 2.2 adds two new outbound NAT modes—Hybrid and Disable. The “Disable outbound NAT” option provides a quick and

obvious way to disable all source NAT. In previous versions, enabling manual outbound NAT and deleting all the auto-generated NAT rules resulted in a system with no NAT. This often confused the user. The new option offers a straightforward means to disable NAT where pfSense software acts only as a router for strictly private address spaces, or public IPs.

Hybrid outbound NAT mode retains the behavior of automatic outbound NAT, but allows configuration of specific outbound NAT rules which are evaluated before the automatically generated rules. One of the more common scenarios where this is helpful occurs with VoIP PBXes requiring the use of static port. Static port disables the rewriting of source ports as part of outbound NAT. In most

Mappings:

	Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description
<input type="checkbox"/>	WAN	192.168.0.0/16	*	*	*	WAN address	*	NO	Internal networks to WAN
<input type="checkbox"/>	WAN2	192.168.0.0/16	*	*	*	WAN2 address	*	NO	Internal networks to WAN2

FIG. 4

ISILON The industry leader in Scale-Out Network Attached Storage (NAS)

Isilon is deeply invested in advancing FreeBSD performance and scalability. We are looking to hire and develop FreeBSD committers for kernel product development and to improve the Open Source Community.



We're Hiring!

With offices around the world, we likely have a job for you! Please visit our website at <http://www.emc.com/careers> or send direct inquiries to annie.romas@emc.com.

EMC²

ISILON

circumstances, VoIP is fine with the defaults; however, with some providers and/or configurations, rewriting the source port will result in no audio or one-way audio on calls. In prior versions, this requirement left one having to manually maintain the entire outbound NAT configuration from that point forward. In 2.2, simply use hybrid outbound NAT, add a single outbound NAT rule per-WAN for the PBX, and leave the remainder at the automatically managed defaults. As an example, consider a scenario with a PBX at 192.168.128.10 on LAN and the need to enable static port for all of its traffic when leaving WAN and WAN2.

As shown in Figure 5, the “WAN static port for PBX” rule applies for all traffic sourced from 192.168.128.10 (a /32 mask in IPv4 specifies a single IP), egressing WAN. Similarly for WAN2 on the “WAN2 static port for PBX” rule. Everything else still falls back to the automatic rules, which will continue to be managed automatically.

Package System

Package signing is well understood in security

circles as vital. Without it, a single mirror server can be compromised, allowing data on that server to be changed. Perhaps a malicious version of a program could be compiled and placed on the server. Any systems using that mirror for updates will download and install that compromised program, with no way to detect modification.

We’ve taken some steps to mitigate the inherent issues in this area, dating back to the project’s early days in only hosting package files on systems we fully control. No mirrors have package files. More recently, we changed to using HTTPS for fetching package files. Now in pfSense version 2.2, packages are pulled only from servers we fully control, only using HTTPS, and must have a valid signature. Our base system stable release updates have always been signed, with the system defaulting to refusing updates without a proper signature.

Translations

Our 2.1 release added gettext support for translations to other languages. The only translation completed in 2.1x releases was

FIG. 5

Port Forward 1:1 Outbound NAT

Mode:

- ☐ Automatic outbound NAT rule generation (IPsec passthrough included)
- ☒ Hybrid Outbound NAT rule generation (Automatic Outbound NAT + rules below)
- ☐ Manual Outbound NAT rule generation (AON - Advanced Outbound NAT)
- ☐ Disable Outbound NAT rule generation (No Outbound NAT rules)

Mappings:

Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description
WAN	192.168.128.10/32	*	*	*	WAN address	*	YES	WAN static port for PBX
WAN2	192.168.128.10/32	*	*	*	WAN2 address	*	YES	WAN2 static port for PBX

Automatic rules:

Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description
WAN	127.0.0.0/8 192.168.128.0/24 192.168.129.0/24	*	*	500	192.0.2.11	*	YES	Auto created rule for ISAKMP
WAN	127.0.0.0/8 192.168.128.0/24 192.168.129.0/24	*	*	*	192.0.2.11	*	NO	Auto created rule
WAN2	127.0.0.0/8 192.168.128.0/24 192.168.129.0/24	*	*	500	198.51.100.15	*	YES	Auto created rule for ISAKMP
WAN2	127.0.0.0/8 192.168.128.0/24 192.168.129.0/24	*	*	*	198.51.100.15	*	NO	Auto created rule

GEOM Mirror information			
Mirror Status	Name	Status	Component
	pfSenseMirror Size: 5368708608	COMPLETE	da0 (ACTIVE) [Rebuild] [Deactivate] [Remove]
			da1 (ACTIVE) [Rebuild] [Deactivate] [Remove]
Some disk operations may only be performed when there are multiple consumers present in a mirror.			
Consumer information			
Available Consumers	No unused consumers found		
Consumers may only be added to a mirror if they are larger than the size of the mirror.			

FIG. 6

GEOM Mirror information			
Mirror Status	Name	Status	Component
	pfSenseMirror Size: 5368708608	DEGRADED [Forget Disconnected Disks]	da0 (ACTIVE)
Some disk operations may only be performed when there are multiple consumers present in a mirror.			
Consumer information			
Available Consumers	Name	Size	Add to Mirror
	da1s1	5362850304 (5.0G)	
	da1	5368709120 (5.0G)	[Reactivate on: pfSenseMirror] [Remove metadata from disk]

FIG. 7

Portuguese. Thanks to community contributors, pfSense version 2.2 adds translations for Japanese and Turkish. We have also recently brought up a translation server at <https://translate.pfsense.org> where we welcome multilingual community members to contribute to translations in other languages.

GEOM Mirrors

The ability to use GEOM mirrors for software RAID has existed in our installer for some time now, and 2.2 brings management enhancements in that area. There is now a diagnostics screen (Figure 6) for GEOM Mirrors in the web interface under Diagnostics>GEOM Mirrors, bringing the ability to manage a software RAID mirror without having to run commands manually via SSH.

One of the requests we've received from users is the ability to break the mirror. This request is often from those who require an additional safety step before they upgrade a system. By clicking the 'Deactivate' link, the mirror is broken, and the upgrade only affects one of the drives.

The "Reactivate" link (Figure 7) will then add the disk back into the mirror and sync it, fully restoring the mirror.

As an additional benefit to this change, to recover after a drive failure, replace the failed drive and reboot the system. After the system has booted, browse to Diagnostics>GEOM Mirrors and click "Forget Disconnected Disks" (Figure 8). (The failed drive will not be returning to the mirror.) Note the status will no longer indicate degraded after doing so since the mirror only has a single member. Clicking the link under "Add to Mirror" (Figure 9) on the new drive will make it the new mirror member.

Click "Confirm" to confirm the addition of the new drive into the mirror. The mirror status will return to a degraded status until the new disk is synchronized. Once finished, the GUI will indicate a normal status.

Monitoring of the mirror's status has also been added. When a mirror is detected as degraded, a notification is filed. As with all system notifications, this can be sent via email and/or Growl notification, using the configuration parameters under System>Advanced, Notifications tab.

FIG. 8

GEOM Mirror information			
Mirror Status	Name	Status	Component
	pfSenseMirror Size: 5368708608	DEGRADED [Forget Disconnected Disks]	da0 (ACTIVE)
			da1 (SYNCHRONIZING, 25%)

GEOM Mirror information

Mirror Status	Name	Status	Component
	pfSenseMirror Size: 5368708608	COMPLETE	da0 (ACTIVE)

Some disk operations may only be performed when there are multiple consumers present in a mirror.

Consumer information

Available Consumers	Name	Size	Add to Mirror
	da1	5368709120 (5.0G)	pfSenseMirror

Consumers may only be added to a mirror if they are larger than the size of the mirror.

WiFi Improvements

FreeBSD 10 also delivered an improved 802.11n WiFi/WLAN wireless networking stack with support for new features and new drivers (e.g., Atheros PCI/PCIe 802.11n WiFi adapter from Qualcomm, SMP/concurrency races, 802.11n TX aggregation).

And Much More...

There have been hundreds of smaller changes across many parts of the system. You can check out the full list here:

https://doc.pfsense.org/index.php/2.2_New_Features_and_Changes

Open-Source Sustainability

There are two views of open-source project sustainability. In one view, for a project to be sustainable it must have contributors, and its codebase needs to be evolving. Users (as volunteers) are also critical to an open-source project with respect to raising requirements and testing. The other viewpoint is that sustainable open-source projects are those that are capable of supporting themselves. Simply put, they are able to meet their ongoing costs. These include infrastructure costs such as hosting and supporting services, and also the costs of developing, updating, and maintaining the codebase. They may also include the costs associated with the governance of the project and with marketing and communications.

For many years, the pfSense project survived on a combination of donations and selling support services. During the past year, we have taken steps to greatly increase the sustainability of the pfSense project. These steps include the addition "pfSense Gold," a premium membership subscription which includes electronic access to the book in PDF, mobi, and epub formats, access to our AutoConfigBackup secure

cloud backup service, monthly hangouts with the pfSense developers, and more. We have also added hardware sales, via store.pfsense.org. Finally, Netgate now directly supports the project by providing nearly all the infrastructure resources, directly employing all the support and development staff, and providing the management and legal resources necessary for a large and still-growing open-source project. •

Jim Thompson has been noodling around the UNIX world for far too long a time. He knows he started with BSD Unix Release 4.0 on a Vax 11/780 in 1980. He still thinks "echo 'This is not a pipe.' | cat - > /dev/tty" is funny. He submitted his first patch to a Free Software project in 1985 for a port of GNU Emacs to a Convex vector supercomputer. Jim refuses to divulge his qualifications and may, in fact, have none at all. He lives in a fortified compound near Austin, Texas with his wife, Jamie, and son, Hunter S. Thompson. His email address is jim@netgate.com.

Chris Buechler is cofounder of pfSense, and its corporate arm, Electric Sheep Fencing, LP. While he was born about the time Thompson started with BSD, he's in his third decade of working professionally with UNIX systems and has been a strong BSD advocate for nearly 15 years. He is one of the authors of the book *pfSense: The Definitive Guide*, the popular and authoritative source of documentation for pfSense, and has presented on security and networking topics at more than 20 conferences in the U.S., Canada, and Europe. He lives in a decidedly non-fortified house in Austin, Texas, with his wife, Sarah, and their two cats. He can be reached at cmb@pfsense.org.

WELCOME

to AsiaBSDCon 2015!

DATE March 12-15, 2015

LOCATION

Tokyo University of Science,
Tokyo, Japan

[www.http://asiabsdcon.org](http://asiabsdcon.org)



CONTACT secretary@asiabsdcon.org

AUTHOR SCHEDULE

November 21, 2014.....Deadline for
Submission of Tutorial Proposals

November 28, 2014.....Deadline for Paper Submission

December 19, 2014.....Notification of
Acceptance of Papers and Tutorial Proposals

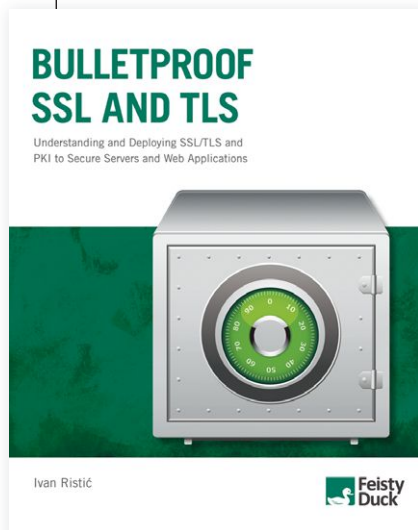
January 9, 2015.....Deadline for Submission of
Tutorial Materials

January 16, 20152015 Deadline for
Submission of Final Papers



BOOKreview

by Steven Kreuzer



Bulletproof SSL and TLS

Understanding and Deploying SSL/TLS and Internet PKI to Secure Servers and Web Applications by Ivan Ristić

Deploying SSL in a sane and secure manner can be a massive challenge for the majority of developers and system administrators. Focusing on SSL since 2009, and dedicating two years to writing, including six months of rewriting and chapter expansion, Ivan Ristić released his second book,

Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications, in August 2014 as the definitive reference to building secure servers and web applications. His exhaustive research and focus have paid off. Ivan Ristić achieves and exceeds his goal “to save you time.”

Bulletproof SSL and TLS is a comprehensive, digestible introduction to cryptographic security, SSL/TLS, and PKI, including common attack methods and dedicated chapters with information on how to securely configure popular platforms such as Apache, Java, and Nginx.

As the Internet evolves and applications become more and more complex, it becomes more and more important to understand the tradeoffs between security, performance, and backwards compatibility. While it would be impossible for Ristić to provide a blueprint for every reader’s individual infrastructure, he does a fantastic job explaining the many available options. Applying his background in application security research and software engineering, he lays out the benefits as well as what you may have to sacrifice in order to assist you in determining the best solution for implementation based on your needs. Ristić further satisfies his audience’s needs and stays true to his goals by acknowledging the rapid evolution of the subject matter and ensuring that chapters will be updated and immediately available to those who

purchased in digital format. After all, TLS is Ristić’s main focus and his vast knowledge base is eagerly shared, ensuring risks are identified and mitigated as much as possible.

Throughout the first half of ***Bulletproof SSL and TLS***, Ristić gives a thorough background on various and relevant subjects while providing straightforward advice. With that in mind, I would suggest if you are currently using TLS to secure communications, then get your hands on a copy of this book and immediately open it up to Chapter 8 on deployment. To quote Ristić, “In many ways, this chapter is the map for the entire book.” Once you have read this chapter, keep a copy by your side while you audit your existing deployment, as it will surely provide assistance when looking for common implementation issues that could result in using weak encryption algorithms that will provide attackers a potential route to compromising your systems. While I was writing this review, a new protocol downgrade attack was discovered in SSLv3. Normally I would have to scramble to verify that all the systems I am responsible for are not vulnerable and fix those that are—this time it was a bit different for me. Since I had taken Ristić’s advice and had already disabled SSLv2 and SSLv3 on all of my internal web servers, all I had to do was inform my manager that because of the changes implemented during my prior audits, we were not at risk and it was business as usual.

After you complete your audit, I suggest you move on to Chapter 9, which focuses solely on performance optimization. When designing your systems you will need to recognize that security is never completely free and no one has ever said they wished their application was slower. Studies have shown that slower page load times will result in an increase in page abandonment. That said, it would seem counterintuitive to devote a tremendous amount of time to profiling your servers and application stack looking to

“

Throughout the first half of *Bulletproof SSL and TLS*, Ristić gives a thorough background on various and relevant subjects while providing straightforward advice. With that in mind, I would suggest if you are currently using TLS to secure communications, then get your hands on a copy of this book and immediately open it up to Chapter 8 on deployment. To quote Ristić, “In many ways, this chapter is the map for the entire book.”

”

eliminate as many bottlenecks as possible only to have to enable encryption. As processors have become faster, it is no longer as expensive as it once was, but encryption will always add overhead, which in turn will increase latency. This chapter provides you with functional advice on how to tune your systems to reduce the performance penalty you normally would have to pay when introducing any form of encryption into your application stack. Although the topic is outside the scope of the book, it provides practical advice on optimizing TCP. Unfortunately, the examples shown are Linux specific, but with a little research of your own, it is fairly trivial to figure out the equivalent settings on FreeBSD.

Ristić does a great job dissecting complex subject matter, such as cryptographic operations and public key infrastructure, which are imperative to building secure systems. He breaks down what is extremely vast and tends to be difficult to understand and presents it in an easy-to-digest manner. This understanding is crucial when designing secure systems, and while it can at times seem overwhelming, Ristić provides his readers with relevant information in a nonintimidating manner.

Bulletproof SSL and TLS brings much to the table, for both the novices and the seasoned professionals who are refreshing their skill sets or learning something new. I personally found the information on OpenSSL to be the most helpful. It is quite possible that OpenSSL is one of the most widely deployed pieces of software on the Internet, protecting not only web servers but also email, chat, virtual private networks, and countless

other network appliances. Despite the fact that it is such a critical component to making the Internet work, many have criticized it as being poorly documented and incredibly difficult to use.

Furthermore, in most environments, assuming it appears to be working correctly, it tends to be back-burnered.

Most people are fortunate enough to not work with OpenSSL on a daily basis. It has been my experience that when someone needs to generate a new private key or create a certificate signing request, they quickly google their intentions, find the first undated blog post that documents someone's commands, and run those commands verbatim without having a clear understanding of the results. Chapters 11 and 12 provide incredibly thorough documentation outlining how to implement OpenSSL for key and certificate management, discuss configuration and deployment, and provide instructions on using OpenSSL to test SSL services. Even if you only make moderate use of OpenSSL on the command line, these chapters will save you countless hours of searching the Internet for answers.

Simply put, neither does Ivan Ristić deny that there are flaws in current security protocols and standards, nor does he try to ignore them. Rather he confronts them head on, accepting and working with the faults to find the most secure solution for your infrastructure—a goal that he clearly achieved in a precise, comprehensive, and readable format. I highly recommend *Bulletproof SSL and TLS* to all system administrators, developers, and managers whether you are already using SSL/TLS and PKI or just getting started.

Bulletproof SSL and TLS

By Ivan Ristić

Release dateAugust 1, 2014

Language.....English (528 pages)

ISBN.....978-1907117046

FormatsPaperback, PDF, EPUB,
Kindle, Online (no DRM)

Published byFeisty Duck
contact@feistyduck.com

Steven Kreuzer is a FreeBSD developer and Unix systems administrator with an interest in retrocomputing and air-cooled Volkswagens. He lives in Queens, New York, with his wife and dog.

PORTSreport

by Frederic Culot

Our ports tree saw quite a lot of activity in September and October. But before I tell you about that, I am pleased to let you know that during the last two months there have been 4,694 commits on the ports tree and 1,039 problem reports closed. Again, this shows our volunteers' dedication to keeping the ports tree up-to-date and taking into account our users' feedback. Many thanks to everyone. And now, on to the main events!

IMPORTANT PORTS UPDATES

Several exp-runs were performed (almost 30 actually!) to check whether major port updates are safe or not. Among those important updates, we mention the following highlights:

- default ruby version is now 2.0
- cmake updated to 3.0.2
- gmake updated to 4.1
- qt updated to 5.3.2
- kde updated to 4.14.2
- Linux base switched from Fedora 10 to CentOS 6

As usual, if manual steps are needed to update specific ports, then those steps are clearly mentioned in the `/usr/ports/UPDATING` file. It is highly advisable to check this file before performing any updates on the ports tree.

NEW PORTS COMMITTERS AND SAFEKEEPING

Three new talents were welcomed to the ranks of ports committers.

Dominic Fandrey will be mentored by `cs@` and `koobs@`. Next, **sbruno@**, who already had a src bit, will be mentored by `bdrewery@` and `bapt@`. Last, **Gordon Tetlow** was also granted a ports commit bit and will be mentored by `erwin@` and `mat@`.

Several commit bits were taken in for safekeeping because they had not been used for a long while (typically more than a year). The ports svn repository access is revoked for those developers who have not worked on ports for a period of time. However, their accounts on FreeBSD's infrastructure are maintained, and whenever they want to come back, they can easily be reinstated. Note that it might be necessary to be given a mentor when a developer is reinstated. Depending on the length of the idle period, the ports tree infrastructure might have changed significantly. So here are the developers who saw their commit bit taken in for safekeeping: `sylvio@`, `pclin@`, `flz@`, `jsa@`, `anders@`. We hope to see them back again soon, and thank them for their work so far.

NOTEWORTHY

Changes to the Ports Tree

The ports tree saw a massive cleanup of the pkg-plist files, thanks to the deprecation of the `@dirrm` and `@dirrmtry` keywords. This cleanup was possible because `pkg(8)` is now able to automatically remove the directories under `${PREFIX}` when needed. As a result, the majority of the lines found in pkg-plist files that contained `@dirrm/@dirrmtry` directives were suppressed. However, please note that it might still be necessary to specify directories to be cleaned up (mainly those created outside of `${PREFIX}` such as game files created in `/var/games/`), in which case the new `@dir` keyword must now be used.

Please note that the `@cwd` keyword was also deprecated. All those changes to the ports infrastructure are documented, and both developers and users might refer to two important sources to keep track of the changes. The first is the porters' handbook (<https://www.freebsd.org/doc/en/books/porters-handbook/>) which explains in detail the usage of new keywords as soon as they appear. The second is the `/usr/ports/CHANGES` file, which is sometimes overlooked, but which contains important technical details mainly relevant to ports committers, but which could also enlighten end users. So everyone is welcome to check this file from time to time.

Frederic Culot has worked in the IT industry for the past 10 years. During his spare time he studies business and management and just completed an MBA. Frederic joined FreeBSD in 2010 as a ports committer, and since then has made around 2,000 commits, mentored six new committers, and now assumes the responsibilities of portmgr-secretary.

LEAK THIS!

• BLOG IT • TWEET IT • READ IT



FreeBSD Journal is available as a mobile app for iPad, iPhone, Kindle, and all Android devices. Also available as a desktop Edition at www.freebsd.foundation.org. \$19.99 per year (6 issues), or \$6.99 per single copy.

conference **REPORT**

by Shteryana Shopova



Grace Hopper Celebration of Women in Computing

I had heard of The Grace Hopper Celebration of Women in Computing, but only had a vague idea of what to expect after Dru Lavigne asked me to come help with the FreeBSD Foundation booth. The world's largest gathering of women technologists was held on October 8–10, 2014, in the Phoenix Convention Center, Phoenix, Arizona, USA.

The event reportedly attracted around 8,500 participants, most of whom were women interested in technology, with attendees ranging from professors from the most renowned universities to senior level officers at large tech companies to university students and graduates seeking career opportunities. The conference activities included panel sessions, an exhibition hall and job fair, an open-

source day panel, poster sessions and plenaries, a film festival, a student opportunity lab, and probably a whole lot more. The event was so big and diverse that it was impossible for one person to grasp all of it.

The conference opened with a wonderful keynote by Shafi Goldwasser, the RSA Professor of Electrical Engineering and Computer Science at MIT and 2012 ACM A.M. Turing Award Winner. She talked about cryptography and how the focus has shifted from wartime uses in the past to privacy of computation and maintaining enough security while still allowing any necessary data to be used nowadays. I did not personally attend the talk since we had a long day ahead, but because the key sessions were live-streamed and recorded, I could catch parts of it before we left the hotel for the venue and at home after the event

(<http://gracehopper.org/2014-grace-hopper-celebration-wednesday-livestream/>).

Once at the venue, we made sure to find our way to the Exhibition Hall and Career Fair, which is where the sponsor booths were located and where the Grace Hopper 2014 Open Source Day

sessions were held. The Open Source Day started with a talk from Shauna Gordon-McKeon from OpenHatch titled “How to Contribute to Open Source.” She gave an overview of how open source projects function—from communication and collaboration to the first steps in contributing to an open-source project. Next up was Dru Lavigne with “An Introduction to the FreeBSD Project,” a talk that provided insight into the specific FreeBSD areas of focus, including file systems, networking, and security. The talks were well attended and I was happy to see so many young university students curious about open source. After the talks I had a very interesting conversation with a senior member of the Python community about mentoring young contributors/developers. I am happy with the policy the FreeBSD Project has adopted for guidance and mentoring new committers and she was also definitely very impressed.

The Exhibition Hall was huge and there were more than a hundred—possibly two hundred—booths with all the big companies represented—Google, Netflix, NetApp, Yahoo, Juniper, to name a few—and many universities—UC Berkeley, MIT, Carnegie Mellon, etc.—all present. Even the NSA had a booth! The FreeBSD Foundation booth was conveniently located next to the area where the coffee and food were served, so we had many students who were just passing by to get coffee and a snack stop at the booth and ask about FreeBSD. This is the first time the FreeBSD Foundation was at the Grace Hopper Celebration, and Dru and I discussed how to possibly improve the FreeBSD Foundation's presence at future Grace Hopper Celebrations. If you were there and have ideas about that, please let us know!

The event did not get by without some controversy. During the “Men Allies” panel, Microsoft’s CEO, Satya Nadella, suggested that women should not ask for a raise, but instead trust the HR team’s judgments. After maddening the conference audience and people on social networks, Nadella later apologized via email to Microsoft employees and publicly on Twitter. Nadella’s misstep was widely covered in the news, so you’ve probably read about it.

We spent a lot of time at the FreeBSD Foundation booth just talking to people. I brought along my copy of the second edition of *The Design and Implementation of the FreeBSD Operating System* and recommended it to students who expressed interest in gaining more insight into operating systems. It was also encouraging to have people who work for companies like Netflix, NetApp, Juniper, Cisco, etc.—those who use FreeBSD—drop by the booth and say, “Hey, we use it” or “Thank you for FreeBSD!”

The Grace Hopper Celebration organizers tried something new this year, calling it “lunchtime table topics.” The idea was that people who had an interest in a specific topic would sit down, have lunch and discuss the topic informally with a moderator who was knowledgeable on the topic and could guide the conversation. Tables with topics like “How to Get Involved with Open Source” or “How to Find a Mentor” attracted the most attention.

Gender diversity in IT has been a huge problem for years, and it is great that events like the Grace Hopper Celebration raise awareness, discuss problems, and propose solutions to the particular problems faced by women in technology. It was definitely a very interesting experience. •

Shteryana Shopova is a software developer specializing in embedded and networking applications. She joined the FreeBSD development team in 2006, and has been promoting FreeBSD and open source software in various ways, including mentoring some Google Summer of Code projects and organizing a local conference dedicated to open source software in her home city of Sofia, Bulgaria. In 2014 she was the Organizing Committee Chair for EuroBSDCon, the largest annual conference in Europe dedicated to the BSD family of operating systems.

Let **FreeBSD Journal**
connect you with a
targeted audience!

Advertise Here

**CLIMB
WITH US!**

→ **LOOKING
for qualified
job applicants?**

→ **SELLING
products
or services?**



Email
walter@freebsdjournal.com

OR CALL
888/290-9469

svn UPDATE

by Glen Barber

THE SVN UPDATE COLUMN NORMALLY CONTAINS INFORMATION ON FEATURES BEING ADDED TO FREEBSD THAT WOULD APPEAR IN AN UPCOMING RELEASE.

This edition, however, overlaps a bit with a release. In lieu of outlining the features that are due to appear in an upcoming release, Glen Barber, on behalf of the FreeBSD Release Engineering Team, would like to thank the FreeBSD community and the FreeBSD developers for all the hard work that went into the FreeBSD 10.1-RELEASE.

Please be sure to browse through the FreeBSD 10.1-RELEASE notes, available at

<https://www.freebsd.org/releases/10.1R/relnotes.html>

as well as the PGP-signed announcement email at

<https://www.freebsd.org/releases/10.1R/announce.asc>

which contains the SHA256 and MD5 checksums of the images.

As a hobbyist, Glen Barber became heavily involved with the FreeBSD project around 2007. Since then, he has been involved with various functions, and his latest roles have allowed him to focus on systems administration and release engineering in the Project. Glen lives in Pennsylvania, USA.

2.3. Userland changes. A new flag is added to camcontrol(8), -b, which outputs the existing buses and their parents. [r260] The newsyslog(8) utility has been updated based on the actual file size on disk. This matches



FreeBSD JOURNAL

NOW AVAILABLE AS A
DYNAMIC EDITION!



The Dynamic Edition format offers subscribers the same features as the App, but permits viewing the Journal through your favorite browser.



Read It

Today!

The DE, like the App, is an individual product. You will get an email notification each time an issue is released.

A one-year subscription is \$19.99, and a single copy is \$6.99—the same pricing as the App version.

**www.
freebsd.foundation.com**

THE INTERNET NEEDS YOU

GET CERTIFIED AND GET IN THERE!

Go to the next level with



Getting the most out of
BSD operating systems requires a
serious level of knowledge
and expertise ● ● ● ● ● ● ● ●

SHOW YOUR STUFF!

Your commitment and
dedication to achieving the
BSD ASSOCIATE CERTIFICATION
can bring you to the
attention of companies
that need your skills.

NEED AN EDGE?

● **BSD Certification can
make all the difference.**

Today's Internet is complex.
Companies need individuals with
proven skills to work on some of
the most advanced systems on
the Net. With BSD Certification

**YOU'LL HAVE
WHAT IT TAKES!**

BSDCERTIFICATION.ORG

Providing psychometrically valid, globally affordable exams in BSD Systems Administration



this month

In FreeBSD

BY DRU LAVIGNE

FreeBSD turned 21 this year, as the first RELEASE announcement went out on November 1, 1993. You can read the announcement for FreeBSD 1.0-RELEASE at <https://www.freebsd.org/releases/1.0/announce.html>. Jordan Hubbard, the release engineer for 1.0, gave a presentation entitled “FreeBSD: The Next 10 Years” at MeetBSD, held in San Jose on the 21st anniversary date. A recording of that presentation is available at <https://archive.org/download/bsdtalk247/bsdtalk247.ogg>.

November of this year also saw its own release announcement as FreeBSD 10.1 was released on November 14. This month's column takes a look at some of the features of 10.1-RELEASE, divided by interest.

For Desktop Users:

The long-awaited, new console driver, vt(4), has been added. This driver provides support for Unicode UTF-8 text with double-width characters and large font maps, including support for Asian character sets. Integration with KMS means that the ability to use Ctrl Alt Fx to scroll through virtual terminals now works again.

Initial support for UEFI boot, including serial console and null console support, has been added for the FreeBSD/amd64 architecture. UEFI images containing this support are available for download from the FreeBSD website.

The system automount facility has been replaced by autofs(5). This new automounter is a clean-room implementation that addresses the limitations of the old automount system. It provides functionality available in most other Unix systems, using proper kernel support implemented using an autofs filesystem. The new automounter integrates with the Lightweight Directory Access Protocol (LDAP) service and has been tested in several enterprise and university environments with thousands of map entries.

Support for UDP Lite has been added. This connectionless protocol allows packets with potentially damaged data to be delivered to an application rather than be automatically discarded. This allows decisions about the integrity of the data to be made by the application or the codec. It is designed for multimedia protocols, such as streamed video, where receiving a packet with a damaged payload is better than receiving no packet at all.

For Virtualization Users:

Support for several VAAI (vStorage APIs for Array Integration) primitives was added to the CAM target layer, ctl(4). VAAI is an API framework that enables certain storage tasks, such as thin provisioning, to be offloaded from the virtualization hardware to the storage array. The following additions greatly improve support for VMware VAAI acceleration, Microsoft ODX (Offloaded Data Transfer) acceleration, and Windows 2012 clustering:

unmap: tells ZFS that the space created by deleted files should be freed. Without unmap, ZFS is unaware of freed space made using a virtualization technology such as VMware or Hyper-V.

atomic test and set: allows a virtual machine to only lock the part of the virtual machine it is using rather than locking the whole LUN, which would prevent other hosts from accessing the same LUN simultaneously.

write same: when allocating virtual machines with thick provisioning, the necessary write of zeroes is done locally, rather than over the network, so virtual machine creation is much quicker.

xcopy: similar to Microsoft ODX, copies happen locally rather than over the network.

The BSD Hypervisor bhyve(8) is a Type-2 hypervisor that supports a number of guests, including FreeBSD, OpenBSD, NetBSD, and several Linux distributions. This RELEASE saw several improvements to bhyve:

- Support for booting from the ZFS filesystem.
- Soft power-off functionality via the acpi(4) S5 state.
- Virtualized XSAVE support to permit guests to use XSAVE-enabled features like AVX (Advanced Vector Extensions).

Several options have been added to bhyve(8): -U allows the UUID of the guest to be specified, -e sets loader(8) environment variables, -C specifies the guest console device, and -H passes the host path to bhyveload(8).

FreeBSD's implementation of VirtIO, virtio(4), provides a BSD-licensed, clean-room implementation of the paravirtualization interface developed for the Linux Kernel-based Virtual Machine (KVM). This RELEASE added several improvements to VirtIO:

- The API has been expanded from 32- to 64-bit.
- The virtio_blk(4) and virtio_scsi(4) drivers have been updated to support unmapped I/O. Unmapped I/O greatly reduces latency and increases I/O scalability and performance on multi-processor systems.
- The virtio_random(4) driver has been added, allowing FreeBSD virtual machines to harvest entropy from the hypervisor.

For ARM Users:

Support for the ARM architecture continues to improve. In 10.1, support was added for the following:

- CHROMEBOOK (Samsung Exynos 5250)
- COLIBRI (Freescale Vybrid)

- COSMIC (Freescale Vybrid)
- IMX53-QSB (Freescale i.MX53)
- QUARTZ (Freescale Vybrid)
- RADXA (Rockchip rk30xx)
- WANDBOARD (Freescale i.MX6)

The following drivers have been added to support TI platforms, such as BEAGLEBONE and PANDABOARD:

- PRUSS (Programmable Realtime Unit Subsystem)
- MBOX (Mailbox hardware)
- SDHCI (new faster driver for MMC/SD storage)
- PPS (Pulse Per Second input on a GPIO/timer pin)
- PWM (Pulse Width Modulation output)
- ADC (Analog to Digital Converter)

Expect more improvements in subsequent FreeBSD releases as well as support for the emerging ARMv8 architecture. For example, the FreeBSD Foundation and Cavium Inc. recently announced their collaboration for creating FreeBSD Tier 1 recognition of the ARMv8 architecture along with an optimized implementation for the Cavium ThunderX processor family. The press release is available from <http://www.prnewswire.com/news-releases/cavium-to-sponsor-freebsd-armv8-based-implementation-277724361.html>.

For ZFS Users:

The bookmarks feature flag has been added. Bookmarks mark the point in time when a snapshot was created and can be used as the incremental source

for a "zfs send" command.

The `vfs.zfs.min_auto_ashift` sysctl can be used to set the minimum ashift value when creating new top-level virtual devices on Advanced Format drives.

The libzfs thread pool API has been imported from OpenSolaris and adapted for FreeBSD. This allows parallel disk scanning which can reduce pool import times for some workloads.

The `restore(8)` utility has been updated to prevent assertion failures when restoring a UFS filesystem dump to a ZFS filesystem.

The default ARC hash table size has been increased and a new loader(8) tunable, `vfs.zfs.arc_average_block-size`, has been added.

The FreeBSD installer, `bsdinstall(8)`, has been updated to include optional GELI-encrypted or mirrored swap devices when installing onto a ZFS filesystem.

Additionally, the parent ZFS dataset is now configured with LZ4 compression enabled.

For More Information:

We've only covered a fraction of some of the new features available in FreeBSD 10.1-RELEASE. For more information, refer to the Release Announcement and the Release Notes:

- <https://www.freebsd.org/releases/10.1R/announce.html>
- <https://www.freebsd.org/releases/10.1R/relnotes.html>

Dru Lavigne is a Director of the FreeBSD Foundation and Chair of the BSD Certification Group.

FreeBSD Journal

The **NEW** publication for FreeBSD buffs, is **OLD** enough now to have

5 BACK ISSUES!



read all you missed!

Order as desktop editions at www.freebsd.foundation.org or purchase at your favorite app store.



Events Calendar

The following BSD-related conferences are scheduled for the first quarter of 2015. More information on these events, as well as local user group meetings, can be found at bsdevents.org.



FOSDEM • Jan. 31 – Feb. 1 • Brussels, Belgium

<https://fosdem.org/2015/> • FOSDEM is a free event that offers open-source communities a place to meet, share ideas, and collaborate. This annual event attracts over 5,000 attendees each year. This year's event features a BSD devroom, a FreeBSD booth in the expo area, and an opportunity to take the BSDA certification exam.



SCALE • Feb. 19 – Feb. 22 • Los Angeles, CA

<http://www.socallinuxexpo.org/scale13x/> • The 13th annual Southern California Linux Expo will once again provide several FreeBSD-related presentations, a FreeBSD booth in the expo area, and an opportunity to take the BSDA certification exam. This event requires registration at a nominal fee.

GREAT WIDE OPEN • March 3 & March 4 • Atlanta, GA

<http://greatwideopen.org/> • This annual technical conference explores open source in the enterprise. There will be a FreeBSD booth in the expo area. Registration is required for this event.



AsiaBSDCon • March 12 – March 15 • Tokyo, Japan

<http://2015.asiabsdcon.org/> • This is the annual BSD technical conference for users and developers on BSD-based systems. It provides several days of workshops, presentations, a Developer Summit, and an opportunity to take the BSDA certification exam. Registration is required for this event.



Are you aware of a conference, event, or happening that might be of interest to *FreeBSD Journal* readers?

Submit calendar entries to editor@freebsdjournal.com.



**Testers, Systems Administrators,
Authors, Advocates, and of course
Programmers** *to join any of our diverse teams.*

COME JOIN THE PROJECT THAT MAKES THE INTERNET GO!

★ **DOWNLOAD OUR SOFTWARE** ★

<http://www.freebsd.org/where.html>

★ **JOIN OUR MAILING LISTS** ★

<http://www.freebsd.org/community/maillinglists.html?>

★ **ATTEND A CONFERENCE** ★

